THESIS

AN EFFICIENT IMPLEMENTATION OF A BATCH-ORIENTED, MULTITARGET, MULTIDIMENSIONAL ASSIGNMENT TRACKING ALGORITHM WITH APPLICATION TO PASSIVE SONAR

by

Sunil Mathews

March 2011

Thesis Advisors:	Tod E. Luginbuhl
Second Reader	Robert G. Hutchins

13. ABSTRACT (maximum 200 words)

This research investigates the use of two versions of a batch-oriented, multidimensional assignment tracking algorithm to examine target crossings that are on the order of 100 scans in duration. The simulations use outputs in one dimension (bearings only) from a passive sonar line array. Linear programming relaxation is used to solve the assignment problem for an exhaustive set of measurement-to-track N-tuple costs along the batch. The implementation of the cost evaluations used for the objective function is analyzed for efficiency. The objective function is minimized subject to certain constraints. The constraints are set up such that each measurement-to-track assignment is exclusive per scan along the batch. The algorithm is generic and can be extended to N dimensions (ND). Missing measurements are accounted for as part of the assignment model. An efficient version of the ND assignment is developed to increase the batch length for acceptable runtime performance. Batch lengths of up to 15 scans, equivalent to a 16D assignment, have been developed and tested on various levels of clutter data. Results are tested via 100-trial Monte Carlo simulations for the two algorithms as applied to the long-duration passive sonar crossing targets case with various clutter density and filter settings.

14. SUBJECT TERMS Batch Tracking, Clutter, Kalman Filter, Multidimensional Assignment, Multitarget Tracking, Passive Sonar, Tracking			**15. NUMBER OF PAGES** 81
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

AN EFFICIENT IMPLEMENTATION OF A BATCH-ORIENTED, MULTITARGET, MULTIDIMENSIONAL ASSIGNMENT TRACKING ALGORITHM WITH APPLICATION TO PASSIVE SONAR

Sunil Mathews
Civilian, Naval Undersea Warfare Center Division, Newport
B.E.E.E., City College of New York, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

from the

NAVAL POSTGRADUATE SCHOOL
March 2011

Author: Sunil Mathews

Approved by: Tod E. Luginbuhl
Thesis Advisor

Robert G. Hutchins
Second Reader

Daphne Kapolka
Chair, Engineering Acoustics Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This research investigates the use of two versions of a batch-oriented, multidimensional assignment tracking algorithm to examine target crossings that are on the order of 100 scans in duration. The simulations use outputs in one dimension (bearings only) from a passive sonar line array. Linear programming relaxation is used to solve the assignment problem for an exhaustive set of measurement-to-track N-tuple costs along the batch. The implementation of the cost evaluations used for the objective function is analyzed for efficiency. The objective function is minimized subject to certain constraints. The constraints are set up such that each measurement-to-track assignment is exclusive per scan along the batch. The algorithm is generic and can be extended to N dimensions (ND). Missing measurements are accounted for as part of the assignment model. An efficient version of the ND assignment is developed to increase the batch length for acceptable runtime performance. Batch lengths of up to 15 scans, equivalent to a 16D assignment, have been developed and tested on various levels of clutter data. Results are tested via 100-trial Monte Carlo simulations for the two algorithms as applied to the long-duration passive sonar crossing targets case with various clutter density and filter settings.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

2D	Two-dimensional
c	Cost
$\mathbf{C}$	Kalman filter measurement matrix
D	Dimensional
JPDA	Joint Probabilistic Data Association (algorithm)
KF	Kalman filter
L	Number of tracks
l_n	Number of 2D LP filtered measurements
LP	Linear programming
m	Varying number of measurements per scan
M	Fixed number of measurements per scan
MDA	Multidimensional assignment
MHT	Multi-hypothesis tracker
MTT	Multitarget tracking
N	Number (dimensions)
NEES	Normalized estimation error squared
ND	N-dimensional
No.	Number
NP	Nondeterministic polynomial time
P_D	Probability of detection
PDA	Probabilistic data association
PMHT	Probabilistic multi-hypothesis tracker
ρ	Binary decision variable
RMSE	Root-mean-square error
SNR	Signal-to-noise ratio
TMA	Target motion analysis

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I express my deepest gratitude to Dr. Tod E. Luginbuhl of the Naval Undersea Warfare Center (NUWC) Division, Newport, for guiding me through this research and the writing of this thesis. His valuable insights, contributed in both formal exchanges and informal conversations, led to many of the significant developments in this work. Dr. Luginbuhl and Dr. Peter K. Willet from the University of Connecticut were instrumental in setting up the experimental problem that laid the foundation for my research. Dr. Robert A. LaTourette, Dr. Christian G. Hempel, Mr. David J. Pistacchio, Dr. Evangelos H. Giannopoulos, and Dr. Phillip L. Ainsleigh of NUWC Division Newport provided constant encouragement and technical support. Dr. Mary H. Johnson, also of NUWC Division Newport, provided a valuable and much appreciated critique. I appreciate the financial support of this applied research project by the NUWC Division Newport In-House Laboratory Independent Research and Independent Applied Research Program review committee, chaired by Mr. Richard G. Phillips and Dr. Anthony A. Ruffa. I am also grateful to many other colleagues at NUWC who created a very nurturing and stimulating environment in which to work. The Naval Postgraduate School provided an excellent grounding in underwater acoustics, and Professor Daphne Kapolka, Chair, Engineering Acoustics Academic Committee, assisted me in finding the right advisor and in meeting all the thesis submission requirements. I thank Professor Robert G. Hutchins, in the Department of Electrical and Computer Engineering, for his careful review of this thesis. Most importantly, I thank my family for their loving encouragement and patience throughout this effort.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This study investigates the implementation of two versions of a batch-oriented, multidimensional assignment (MDA) approach for tracking contacts with long-duration crossings, on the order of 100 scans for a single dimensional space. The environment being simulated is that of the passive sonar string in a line array with bearings-only information. The model incorporates multitarget tracking (MTT) in a cluttered environment with missing measurements. Linear programming (LP) relaxation is used to solve the cost assignment matrix. The assignment costs are calculated via the Kalman filter likelihood function. A constraint matrix is set up for the various batch lengths, and the assignment problem is solved via an LP package. This study concentrates on using efficient techniques to eliminate many of the redundancies in generating the cost assignment matrix, and a single-scan, forward-looking filtering method is developed to further reduce the number of generated cost evaluations.

The algorithm is generic and can be extended to the N dimensional (ND) assignment problem. The term ND refers to a number N of dimensions D where the first dimension refers to the tracker state estimates and covariance from the previous scan, and $(N - 1)$ dimensions refers to the batch length in time for the subsequent scans. To minimize computational requirements, a suboptimal version of the algorithm was developed and tested. The suboptimal version is a faster implementation of the ND assignment algorithm, and results have been processed up to 16D. Simulated results have been processed up to 4D for the standard implementation of the algorithm. The efficient implementation utilizes (1) a two-dimensional (2D) LP module as a filtering procedure to extract measurements in clutter and (2) a skip factor of the length of the batch in processing the ND assignment. The standard implementation processes the ND assignment and keeps only the first scan's measurement of the batch, with the procedure repeated for consecutive scans via a sliding batch.

Comparisons are performed for various cases. Test data are generated in a cluttered environment via simulation and processed with various clutter parameter and batch settings. In this study, only simulations with linear motion model targets were considered, and the test data simulate detections obtained in a passive sonar environment for target crossings. Monte Carlo runs are conducted for 100 trials for the various cases. The results show that the suboptimal but fast implementation tracked correctly through long-duration crossings in a much larger percentage of the trials—over 90%—than the standard implementation—less than 30% of the trials.

A. BACKGROUND

A passive line array in a typical ocean environment detects large vessels such as tankers and freighters at a long distance, so that contact information is generally "bearings only." When two of these distant contacts cross in bearing, the crossing duration can be on the order of minutes. During the crossing, if the contacts are not of approximately equal signal-to-noise ratio (SNR), detections will be assigned to the higher SNR contact, and the lower SNR track will be lost. For contacts that have equal probability of detection (P_D), Willett et al. [1] have recently shown that the estimated tracks generated with a standard assignment model exhibit a repulsion behavior during the crossing that violates the expected behavior of these large contacts, i.e., that they are known to travel at constant course and speed and cannot make sudden changes because of their size. This repulsion behavior causes problems downstream for algorithms that require state estimates to be consistent with contact trajectory.

This study concentrates on utilizing a batch-style algorithm to track targets with equal P_D in a long-duration crossing. MDA techniques (generally batch-oriented) are used to investigate tracking of long-duration crossings. Willet et al. [1] explored the use of batch-oriented tracking algorithms for the linear crossing case. The algorithms discussed in this thesis can be generalized to N dimensions. Solving this N-dimensional batch assignment is considered to be "NP hard" (where NP means nondeterministic polynomial time) for the case in which the dimensional value $N \geq 3$; therefore, linear programming relaxation techniques are used. An off-the-shelf software package (LP_SOLVE) [2], [3] is used to solve the objective function for the assignment problem. Three algorithms—a modified probabilistic multi-hypothesis tracker (PMHT), a multi-hypothesis tracker (MHT), and an MHT with rollout—were investigated and tested in [1], and those simulations and test results are the foundation for the simulated results in this study.

In the passive sonar environment, the contact detections from a line array are processed in sequence by a beamformer and a detector and subsequently sent to postprocessing algorithms such as tracking and classification algorithms. Tracks are then formed for the postdetection data and sent to other postprocessing algorithms that require tracks as inputs, such as target motion analysis (TMA) algorithms [4], [5], [6]. In the scope of this thesis, simulations are developed that represent the postbeamformer output of a detector for a line array of hydrophones.

A beamformer [7], [8] is essentially a transformation from the hydrophone's or element's "time" space to "bearing" space (i.e., beam angle or conical) space. In this thesis, a line array of elements is considered. A beamformer from a line array of elements produces a fixed number of beams, based on directional angles (bearings), by delaying and summing the time series from each element. A line array has only a single dimension for its observation space, which is its conical beam space. The outputs of the beamformer are sent to a detection process that will filter and normalize the data to eliminate the unneeded frequency ranges and preserve the ranges of interest. The detector outputs are also assumed to be normalized spatially to have a constant noise background. It is assumed in this study that the outputs of the detector are peak-picked. A peak-picked value is any measurement in the detector output above a preselected threshold. Because the beamformer has a fixed number of beams, the peak-picked values from the detector provide only discrete measurements. These discrete measurements are interpolated based on amplitude or other criteria to provide a finer estimate of the measurements at each scan. These fine measurements will be sent to a tracking algorithm.

The crossing target case considered is the long-duration crossing following a linear trajectory. Test results are processed via Monte Carlo simulation for various clutter levels and various batch lengths.

B. REVIEW OF BATCH-ORIENTED MTT ALGORITHMS

Recent developments in MTT and data fusion technologies are all pointing to the need to include batch frames of data in order to achieve optimal performance. The current batch-oriented MTT algorithms include four classes—the PMHT, the MHT, the batch-oriented joint probabilistic data association (JPDA) tracker, and the MDA tracker. The PMHT developed by Streit and Luginbuhl [9], [10] is one batch-oriented algorithm that has shown promise for passive sonar and other applications. However, the PMHT does not directly produce accurate error covariance matrices for the track estimates; a separate computation is required [1]. The MHT, as originally proposed by Reid [11], has a batch-oriented framework where tracks are constructed based on enumeration of all possible measurement-to-track association hypotheses along a batch. As time evolves, the number of tracks grows exponentially, based on new measurement arrivals with each scan. For practical implementation, current MHT implementations are suboptimal [12] because of computational processing and memory limitations associated with using fixed-size batch lengths. To limit the exponential growth of the number of tracks, even with a fixed batch length, *ad hoc* logic is used to prune tracks that are infeasible and meet certain criteria at a subsequent scan. This type of deferred decision logic can only be applied when using a batch-style tracker. The JPDA algorithm [13, pp. 310–319], [14] is a true multitarget tracking algorithm that produces consistent state estimates (i.e., accurate state and covariance estimates), but batch extensions [15], [16], [17] are still very limited (batch lengths of less than three scans) in the current state of the art. Further, the current version of the algorithm is prone to track segmentation, because of the track coalescence effect of PDA style trackers [18]. In PDA-style trackers, the tracks tend to merge, i.e., coalesce, as they cross, which requires initialization of a new track to replace the segment not tracked past the crossing. The MDA algorithm [19], [20] utilizes an optimization framework, and an enumeration of all of the possible measurement-to-track costs is calculated along the batch. Cost minimization is performed via several techniques. Lagrangian relaxation is used in [21], whereas the interior point linear programming is used in [22]. In [2], a mixed integer linear programming (LP) relaxation method is utilized to minimize the assignment cost matrix with an open source solver, LP_SOLVE [3]. The MDA framework is based on 0-1 integer assignments as originally proposed in [23] via 0-1 integer linear programming, where tracks are given discrete assignments with one measurement per scan and are not allowed to share measurements, based on explicit constraints placed on the optimization. An LP-based method to produce mixed integer assignments is explored in [24]. The current work will investigate an implementation of the MDA algorithm that uses linear programming and incorporates longer batch lengths into this framework. The motivation for longer batch lengths is the premise that extending the batch length to cover the tracks for periods before and after the crossing period will aid in the long-duration crossing target problem by providing enough data on the contact positions to better predict the correct trajectories. This algorithm will be analyzed with simulated results related to the passive sonar, long-duration crossing tracks problem. A suboptimal method is also developed to reduce the computational load of the algorithm—by reducing the number of cost evaluations—to further increase the batch length.

This thesis is organized as follows. In Chapter II, the two algorithms are described and the motion model and measurement model assumptions are formulated. An implementation of the MDA algorithm is presented as Algorithm I. The 2D-ND MDA algorithm, which is a more efficient—and therefore faster—version, is presented as Algorithm II. In Chapter III, which presents the results, the two algorithms are applied to the long-duration crossing target problem for a passive sonar line array. Simulation results are presented for various test cases.

THIS PAGE INTENTIONALLY LEFT BLANK

II. ALGORITHM DESCRIPTION

A. MOTION MODEL ASSUMPTIONS

The true positional state vector for a contact at time scan t_{s+1} is given by

$$\mathbf{x}(t_{s+1}) = \mathbf{A}\begin{bmatrix} x(t_s) \\ \dot{x}(t_s) \end{bmatrix}, \tag{1}$$

where t is the time of the scan, s is the scan index of the original data, and **A** is the state transition matrix of the system.

$$\mathbf{A} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \tag{2}$$

where T is the scan period.

B. MEASUREMENT MODEL ASSUMPTIONS

The measurement model consists of received measurements normally distributed about the fractional beam space x.

$$z(t_s) = x(t_s) + w(t_s)\sigma_r\,, \tag{3}$$

$$R = \sigma_r^{\;2}\,, \tag{4}$$

where $z(t_s)$ is the linear measurement position at time scan t_s, w is the white Gaussian random noise at time scan t_s, σ_r is the standard deviation of the measurement noise, and R is the variance of the measurement noise.

The clutter measurements are assumed to have a Poisson random variable distribution with spatial clutter density parameter λ. The contact's probability of detection P_D is fixed, and spatial clutter density parameter λ is varied for the simulated results.

C. ALGORITHM I: MDA ALGORITHM

1. Overview of the MDA Algorithm

An overview of the MDA algorithm is provided as a flowchart in Figure 1. The flow of the modules is based on [19] and [20], except for the LP_Assign section, which is based here on LP relaxation instead of the Lagrangian relaxation technique. In this study, it is assumed that the tracks are already initialized, where prior track estimates are known. Based on the simulated data, the tracks are initiated on the first scan and the algorithm processes the subsequent scans based on the initiated tracks via a sliding batch length ($N - 1$). No track management functions, such as dropping, pruning, or merging tracks, are considered in this study. No new tracks will be initiated subsequently after the first scan. Much of the processing for an MDA-style algorithm takes place in the GenCost and LP_Assign sections of Figure 1. The GenCost section computes the Kalman filter likelihoods for every possible combination of measurement-to-track associations per scan for the entire ($N - 1$) batch length. It also accounts for missing measurements per scan for the entire batch length per track. This study concentrates largely on the GenCost section of the algorithm and eliminates many of the deficiencies of this module. The LP_Assign section performs the actual measurement-to-track assignment via LP relaxation. This assignment can be performed via various methods such as Lagrangian relaxation, interior point linear programming, and integer linear programming [2], [21]-[24]. This study employs linear programming relaxation via an available open source solver, LP_SOLVE [3].

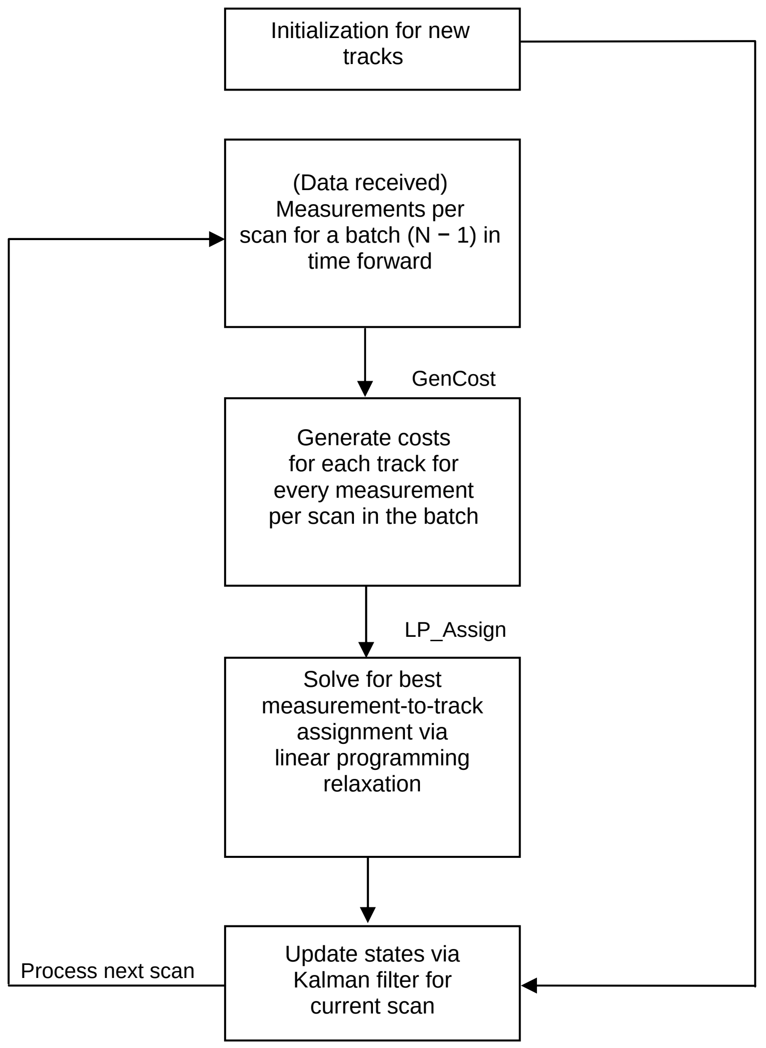
Initialization for new tracks
(Data received) Measurements per scan for a batch (N − 1) in time forward
GenCost
Generate costs for each track for every measurement per scan in the batch
LP_Assign
Solve for best measurement-to-track assignment via linear programming relaxation
Update states via Kalman filter for current scan
Process next scan

Figure 1. Overview of MDA tracking algorithm.

2. Track Initialization

The track initialization function initializes a track based on known input parameters such as initial target (i.e., contact) position $x(t_{s_0})$ and rate $\dot{x}(t_{s_0})$. As part of this study, the tracks are initialized on the initial simulated state estimates based on the motion model assumptions. The rate term $\dot{x}(t_{s_0})$ is set to 0 for the initialized track's state. This function is performed at the first scan.

3. Cost Generation

The generation of the cost evaluations used for the objective function of the MDA algorithm was analyzed for efficiency in this research, and the resulting GenCost module is described here in detail. Cost evaluations are calculated based on all possible measurement-to-track assignment hypotheses including the missing measurement along a batch. The cost evaluations are obtained by the Kalman filter negative log likelihood calculation. For each track, a missing measurement is also accounted for in every scan of the batch. Note that the following prediction equations are used only to compute the likelihoods. At time t_n, the start of each batch, the state estimates and covariance for each track from the prior scan are used and the negative log likelihoods are cumulated along the batch. These prior state estimates and covariances are from the Kalman filter update section of this thesis.

Equations (5) through (13) are from standard Kalman filter theory. The Kalman filter time update comprises the state prediction, given by Equation (5) and the state prediction covariance, given by Equation (6).

$$\mathbf{x}(t_{n+1}) = \mathbf{A}\mathbf{x}(t_n)\,, \tag{5}$$

$$\tilde{\mathbf{P}}(t_{n+1}) = \mathbf{A}\mathbf{P}(t_n)\mathbf{A}' + \mathbf{Q}\,, \tag{6}$$

where the index n is the scan index in the batch, $\tilde{\mathbf{P}}$ is the predicted state prediction covariance, $\mathbf{P}$ is the state prediction covariance in current time, and $\mathbf{Q}$ is the covariance of the discrete-time process as defined in Equation (7).

$\mathbf{P}(0)$ is the initial covariance matrix set at

$$\mathbf{P}(0) = \begin{bmatrix} {\sigma_{f_0}}^2 & \dfrac{{\sigma_{f_0}}^2}{T} \\ \dfrac{{\sigma_{f_0}}^2}{T} & {\sigma_{f_0}}^2 \end{bmatrix},$$

where the Kalman filter's initial function value variance ${\sigma_{f_0}}^2$ is a parameter and is the same for all tracks.

The covariance matrix **Q** is based on a discrete white noise acceleration model from [25, p. 274] and given by Equation (7):

$$\mathbf{Q} = \sigma_q{}^2 \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{bmatrix}. \tag{7}$$

The measurement prediction is given by

$$\hat{z}(t_{n+1}) = \mathbf{Cx}(t_{n+1}), \tag{8}$$

where **C** = [1   0] is the measurement matrix, and σ_q is the standard deviation used in the process noise model.

The Kalman filter measurement update is computed using the innovation covariance estimate $\mathbf{S}(t_{n+1})$, filter gain estimate $\mathbf{K}(t_{n+1})$, measurement residual $\nu(t_{n+1})$, updated state estimate $\mathbf{x}(t_{i+1})$, and the updated state covariance $\mathbf{P}(t_{i+1})$, given by Equationss (9) through (13), respectively.

$$\mathbf{S}(t_{n+1}) = \mathbf{C}\tilde{\mathbf{P}}(t_{n+1})\mathbf{C}' + R, \tag{9}$$

$$\mathbf{K}(t_{n+1}) = \tilde{\mathbf{P}}(t_{n+1})\mathbf{C}'\mathbf{S}(t_{n+1})^{-1}, \tag{10}$$

$$\nu(t_{n+1}) = z(t_{n+1}) - \hat{z}(t_{n+1}), \tag{11}$$

$$\mathbf{x}(t_{i+1}) = \mathbf{x}(t_{n+1}) + \mathbf{K}(t_{n+1})\nu(t_{n+1}), \tag{12}$$

$$\mathbf{P}(t_{i+1}) = \tilde{\mathbf{P}}(t_{i+1}) - \mathbf{K}(t_{n+1})\mathbf{S}(t_{n+1})\mathbf{K}(t_{n+1})'. \tag{13}$$

The cost calculation [24], [26] is performed via the following function for the Kalman filter likelihood Λ:

$$\Lambda(t_{n+1}) = \Lambda(t_n) + \frac{1}{2}\nu(t_{n+1})'\mathbf{S}(t_{n+1})^{-1}\nu(t_{n+1}) + \ln\left[\frac{\lambda_e \left|2\pi\mathbf{S}(t_{n+1})\right|^{\frac{1}{2}}}{P_D}\right]. \tag{14}$$

To process a missing measurement, a penalty is imposed where

$$\Lambda(t_{n+1}) = \Lambda(t_n) - \ln(1 - P_D)\ . \tag{15}$$

Note that λ_e is the assumed clutter density used by the tracking algorithm in Equation (14). The clutter density parameter differs from λ in the measurement model section only in that λ_e is a fixed parameter whereas λ varies with the simulation scenario. Note that, in the results section, the case is examined where λ_e equals λ, i.e., the matched case. If a missing measurement is being processed, only the Kalman time update Equations (5) through (8) and (15) are performed. This set of equations—Equations (5) through (15)—is used to calculate the Kalman filter likelihoods Λ that serve as the costs forming the objective function for the MDA algorithm. The basic computational unit for an MDA algorithm is a single Kalman filter update used to calculate the negative log likelihoods. The details of the negative log likelihood function are described in [24] and [26]. Reference [21] provides further details on the cost calculation methodology used for the MDA algorithm. The calculations of Equations (5) and (6) and (8) through (14) are performed for every track at each scan along the batch for all possible combinations of measurement-to-track associations. For missing measurements, Equations (5), (6), (8), and (15) are calculated. Note, that Equations (14) and (15) are used for processing actual measurements or missing measurements respectively. A single likelihood represents a possible trajectory for a track along a batch. Enumeration of all measurement-to-track (including missing measurement) likelihoods needs to be calculated per track for the entire batch length. The number of costs calculated per track is given by Equation (16) for a fixed number of measurements per scan along a batch.

$$\text{Number of costs } (c) = M^{N-1} \text{ or } \prod_{n=1}^{N-1} M_n\ , \tag{16}$$

where M is the fixed number of measurements per scan n, including the missing measurement, and $(N - 1)$ is the batch length. The total number of costs is given by

$$\text{Total number of costs} = M^{N-1}L \text{ or } L \cdot \prod_{n=1}^{N-1} M_n , \tag{17}$$

where L is the number of tracks.

The total number of Kalman filters (KF) to process for all tracks in a batch is given by

$$\text{Total number of KF} = M^{N-1}L(N-1) \text{ or } L \cdot (N-1) \cdot \prod_{n=1}^{N-1} M_n . \tag{18}$$

The Kalman filters used to compute the likelihoods are the basic unit as far as computational costs for a MDA style algorithm, since it must be performed numerous times to evaluate all possible combinations of measurement-to-track costs. Note, that Equations (16) through (18) apply only to cases with a fixed number of measurements per scan and can be used for rough algorithm loading calculations.

If the number of measurements changes from scan to scan, as normally happens, the cost tree and the number of Kalman filters are calculated using generic equations. The number of costs for any number of measurements and the number of Kalman filters per track are calculated using Equations (19) and (20), respectively.

$$\text{Number of costs } (c) = \prod_{n=1}^{N-1} m_n , \tag{19}$$

$$\text{Number of KF updates} = (N-1)\prod_{n=1}^{N-1} m_n , \tag{20}$$

where m is the varying number of measurements for scan n from 1 to ($N - 1$).

Consider an example of two tracks with a batch length of ($N - 1$) of three scans with three measurements in each scan, as shown in Figure 2. This is a 4D Assignment with a batch length of three. The measurements include the missing measurement, where there are only two actual measurements. The topology on the left represents the track paths for Track 1 and on the right for Track 2. The measurements are labeled from one to nine for the three measurements and three scans of data as shown. Note, that measurements 1, 4, and 7 are labeled as missing measurements. From Equation (16), there are 3^3 or 27 possible paths for this specific example per track. As mentioned in Equation (18), the total number of Kalman filter updates to process is $M^{N-1}L(N-1)$. In this case, it would require 162 Kalman filters to generate a total of 54 cost values c in a single batch for the two tracks.

It is important to note that there is no gating procedure in Algorithm I. In the MDA procedure, as outlined in [19] and [21], there is no gating involved, with every measurement-to-track hypotheses calculated, as is the case for Algorithm I. To improve processing speed of MDA style algorithms gating can be performed as shown in [27], using a clustering technique along the batch. Any type of gating used makes the association problem suboptimal, especially when processing increasing time depths. A suboptimal gating technique based on a 2D-ND MDA algorithm is proposed for faster processing. The gating procedure is described in Section D.3 for Algorithm II.

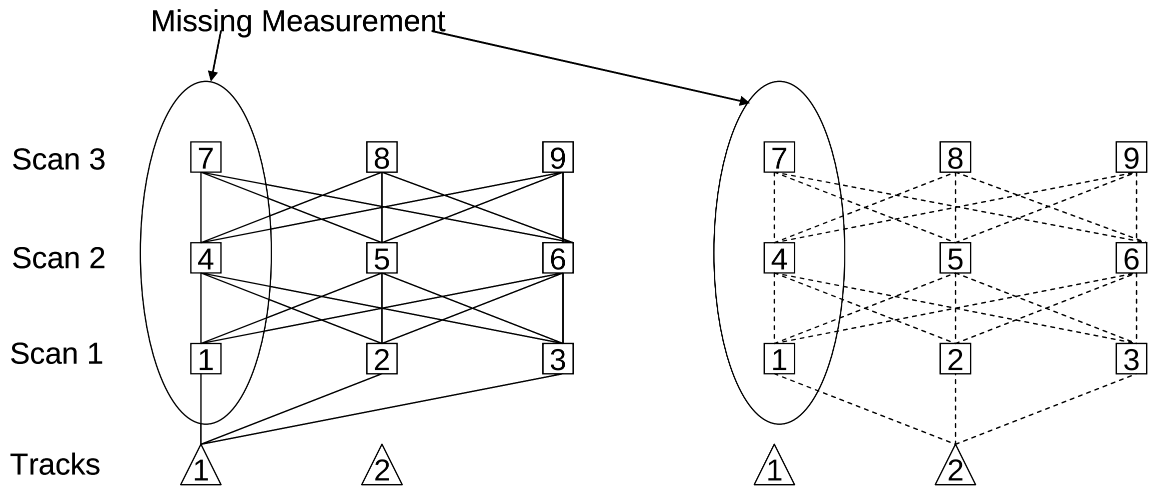

Figure 2. Example of 4D assignment case cost mapping with two tracks and three measurements, including the missing measurement.

In Figure 3, the unraveled track paths are shown for Track 1; the measurement indices in this figure are labeled as in Figure 2. Notice the tree structure of the different paths from the track to the end of the batch. The main trunk of the tree is at the bottom where the tracks start from the first scan, and the canopy, fringe branches are towards the last scan.

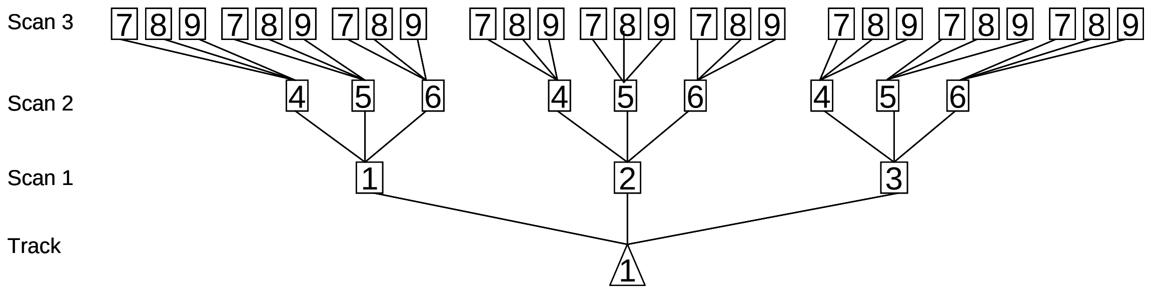

Figure 3. Example of 4D cost tree mapping for Track 1.

Table 1 provides an enumeration of the binary value indexes ρ for obtaining the cost evaluation tree depicted in Figure 3. As previously mentioned, for the 4D example, 27 cost terms c are calculated per track. The top row represents the numbered measurements depicted in Figure 2. The leftmost column represents the associated binary decision values ρ to the costs c. The four subscripted numerals to the decision term ρ represent an N- tuple path along the batch for a given track. In Table 1, the first term in the subscript represents Track 1's costs and associated decision terms ρ, or the index of the first dimension. The second term represents the measurement index in the first scan, the third term represents the measurement index in the second scan, and so on. A Kalman filter calculation is represented by a value of 1 in Table 1, and for this 4D example there are 81 filters used per track.

These costs enforce the underlying assignment model where the constraints imposed require that only one measurement is associated with a track per scan along a batch and that multiple tracks can be associated with the missing measurement at the same scan. This optimization procedure is described for Algorithm I in Section C.5.

Measurement Index

	1	2	3	4	5	6	7	8	9
ρ_{1000}	1			1			1		
ρ_{1001}	1			1				1	
ρ_{1002}	1			1					1
ρ_{1010}	1				1		1		
ρ_{1011}	1				1			1	
ρ_{1012}	1				1				1
ρ_{1020}	1					1	1		
ρ_{1021}	1					1		1	
ρ_{1022}	1					1			1
ρ_{1100}		1		1			1		
ρ_{1101}		1		1				1	
ρ_{1102}		1		1					1
ρ_{1110}		1			1		1		
ρ_{1111}		1			1			1	
ρ_{1112}		1			1				1
ρ_{1112}		1				1	1		
ρ_{1120}		1				1		1	
ρ_{1121}		1				1			1
ρ_{1122}			1	1			1		
ρ_{1200}			1	1				1	
ρ_{1201}			1	1					1
ρ_{1202}			1		1		1		
ρ_{1211}			1		1			1	
ρ_{1212}			1		1				1
ρ_{1220}			1			1	1		
ρ_{1221}			1			1		1	
ρ_{1222}			1			1			1

Table 1. Example of 4D cost enumeration for Track 1: three measurements per scan.

In Table 2, the number of calculated costs *c* for a single track for various *N* dimensional assignments from 2D to 10D assignment and number of measurements from 1 to 10 per scan are provided.

N \ *M*	1	2	3	4	5	6	7	8	9	10
2	1	2	3	4	5	6	7	8	9	10
3	1	4	9	16	25	36	49	64	81	100
4	1	8	27	64	125	216	343	512	729	1000
5	1	16	81	256	625	1296	2401	4096	6561	10000
6	1	32	243	1024	3125	7776	16807	32768	59049	100000
7	1	64	729	4096	15625	46656	117649	262144	531441	1000000
8	1	128	2187	16384	78125	279936	823543	2097152	4782969	10000000
9	1	256	6561	65536	390625	1679616	5764801	16777216	43046721	1E+08
10	1	512	19683	262144	1953125	10077696	40353607	1.34E+08	3.87E+08	1E+09

Table 2. Number of *N*-tuple cost evaluations *c* for various dimensions *N* with a fixed number of measurements *M* per scan.

Table 3 provides the number of Kalman filters needed to generate the costs in Table 2.

N \ *M*	1	2	3	4	5	6	7	8	9	10
2	1	2	3	4	5	6	7	8	9	10
3	2	8	18	32	50	72	98	128	162	200
4	3	24	81	192	375	648	1029	1536	2187	3000
5	4	64	324	1024	2500	5184	9604	16384	26244	40000
6	5	160	1215	5120	15625	38880	84035	163840	295245	500000
7	6	384	4374	24576	93750	279936	705894	1572864	3188646	6000000
8	7	896	15309	114688	546875	1959552	5764801	14680064	33480783	70000000
9	8	2048	52488	524288	3125000	13436928	46118408	1.34E+08	3.44E+08	8E+08
10	9	4608	177147	2359296	17578125	90699264	3.63E+08	1.21E+09	3.49E+09	9E+09

Table 3. Number of Kalman filters needed to generate cost *c* for various dimensions *N* with a fixed number of measurements *M* per scan.

As noted in [27] and [28], the costs calculated using the GenCost function, or generating the Kalman filter negative log likelihood Λ, take up about 95% of the computational requirement of an MDA style of algorithm. In Table 3 for example, with nine measurements and 7D assignment, over three million Kalman filter updates need to be processed for a single track to generate all the costs in the batch. As the number of tracks increase, the cost calculations and the number of Kalman filter updates to process increase linearly with the numbers from Tables 2 and 3, respectively. If the cost generation function is taking up much of the computational resources, then it is beneficial to reduce the number of overall Kalman filter calculations along a batch per track. Some of the cost generation inefficiencies can be eliminated because the Kalman filter negative log likelihoods have previously been calculated for the initial scans of the batch. Storing a select set of previously processed cumulative likelihoods, prior track filter state estimates, and covariances makes these values available for reuse later in the processing, which results in significant computational savings without any effect on the overall algorithm. The circled sections in Table 4 show the costs that are part of the main trunk of the tree that can be reused to calculate the canopy branches of the cost tree from Figure 3. It is observed that the number of links in the cost tree corresponds to the number of Kalman filters to process. The main trunk sections are early in the batch, and the fringe branches, representing the number of cost evaluations, are toward the end of the batch. By using this technique, the number of Kalman filter updates to calculate in the 4D example decreases from 162 to 78 filters for the two example tracks, which is a significant savings without any impact on the integrity of the algorithm. Note that the number of costs calculated for the objective function is still the same. The reduction in the number of Kalman filters provides the computational savings to process the costs.

The equation used to calculate the number of reduced Kalman filter (KF) updates for a fixed number of measurements M in a batch is given by the following special case of the geometric series and generation function, where M > 1:

$$\text{Number of reduced KF} = \sum_{n=1}^{N-1} M^n = \frac{M^N - M}{M-1}. \qquad (21)$$

	Measurement Index								
	1	2	3	4	5	6	7	8	9
ρ_{1000}	1			1			1		
ρ_{1001}	1			1				1	
ρ_{1002}	1			1					1
ρ_{1010}	1				1		1		
ρ_{1011}	1				1			1	
ρ_{1012}	1				1				1
ρ_{1020}	1					1	1		
ρ_{1021}	1					1		1	
ρ_{1022}	1					1			1
ρ_{1100}		1		1			1		
ρ_{1101}		1		1				1	
ρ_{1102}		1		1					1
ρ_{1110}		1			1		1		
ρ_{1111}		1			1			1	
ρ_{1112}		1			1				1
ρ_{1112}		1				1	1		
ρ_{1120}		1				1		1	
ρ_{1121}		1				1			1
ρ_{1122}			1	1			1		
ρ_{1200}			1	1				1	
ρ_{1201}			1	1					1
ρ_{1202}			1		1		1		
ρ_{1211}			1		1			1	
ρ_{1212}			1		1				1
ρ_{1220}			1			1	1		
ρ_{1221}			1			1		1	
ρ_{1222}			1			1			1

Table 4. Example 4D with calculated costs *c*: circled sections are stored cost values.

Even when processing a missing measurement, a Kalman filter time update must be performed to calculate the state estimates for the next scan. The number of Kalman time updates to process missing measurements is given by

$$\text{Number of reduced KF time updates} = \sum_{n=1}^{N-1} M^{n-1} = \frac{M^{N-1}-1}{M-1}. \qquad (22)$$

The total number of full Kalman filters to process real measurements along a batch is given by

$$\text{Number of reduced full KF} = \sum_{n=1}^{N-1}(M^{n}-M^{n-1}) = \frac{M^{N}-M^{N-1}}{M-1}-1 = M^{N-1}-1. \qquad (23)$$

The generic equation to calculate the number of reduced Kalman filters for any number of measurements in a batch, for a single track is given by

$$\text{Number of reduced KF} = \sum_{k=2}^{N}\prod_{n=1}^{k-1} m_n, \qquad (24)$$

where m_n is the number of measurements for scan n from 1 to $(k - 1)$, and k is the dimensional assignment from 2 to N.

Table 5 shows the number of Kalman filters required using the efficient methodology used in this study for N dimensions from 2D to 10D and a fixed number of measurements per scan, from 1 to 10. From Table 3, as was mentioned previously in an example, with nine measurements and 7D assignment, over three million Kalman filters needed to be processed for a single track. In Table 5, with the reduced number of Kalman filters with nine measurements and 7D assignment, the number of Kalman filters dramatically decreases to less than 600,000—a more than fivefold savings.

Table 6 shows the savings in computation achieved with the reduced set of Kalman filters based on the efficient storage methodology. This table gives the ratio of the number of Kalman filters for each case in Table 3—the full set—to the number of Kalman filters required for the same case in Table 5—the reduced set. As is expected, as the N dimensional size and the number of measurements per scan increase, the computational savings significantly increase. In the 10D case with 10 measurements per scan, there is an eightfold decrease in the number of Kalman filters computed.

N \ *M*	1	2	3	4	5	6	7	8	9	10
2	1	2	3	4	5	6	7	8	9	10
3	2	6	12	20	30	42	56	72	90	110
4	3	14	39	84	155	258	399	584	819	1110
5	4	30	120	340	780	1554	2800	4680	7380	11110
6	5	62	363	1364	3905	9330	19607	37448	66429	111110
7	6	126	1092	5460	19530	55986	137256	299592	597870	1111110
8	7	254	3279	21844	97655	335922	960799	2396744	5380839	11111110
9	8	510	9840	87380	488280	2015538	6725600	19173960	48427560	1.11E+08
10	9	1022	29523	349524	2441405	12093234	47079207	1.53E+08	4.36E+08	1.11E+09

Table 5. Number of Kalman filters needed in reduced set where previously calculated costs *c* and filter outputs are stored for various dimensions *N* with a fixed number of measurements *M* per scan.

N \ *M*	1	2	3	4	5	6	7	8	9	10
2	1	1	1	1	1	1	1	1	1	1
3	1	1.33	1.50	1.60	1.67	1.71	1.75	1.78	1.80	1.82
4	1	1.71	2.08	2.29	2.42	2.51	2.58	2.63	2.67	2.70
5	1	2.13	2.70	3.01	3.21	3.34	3.43	3.50	3.56	3.60
6	1	2.58	3.35	3.75	4.00	4.17	4.29	4.38	4.44	4.50
7	1	3.05	4.01	4.50	4.80	5.00	5.14	5.25	5.33	5.40
8	1	3.53	4.67	5.25	5.60	5.83	6.00	6.13	6.22	6.30
9	1	4.02	5.33	6.00	6.40	6.67	6.86	7.00	7.11	7.20
10	1	4.51	6.00	6.75	7.20	7.50	7.71	7.88	8.00	8.10

Table 6. Ratio of number of Kalman filters calculated for full set (per Table 3) vs reduced set (per Table 5). In both sets, *N* is the dimensional size, and there is a fixed number of measurements *M* per scan.

The storage of previously calculated costs *c* and Kalman filter outputs is used in this version of MDA Algorithm I and in the suboptimal but faster Algorithm II.

4. Linear Programming Overview

Once the costs c are obtained, based on an enumeration of N-tuple measurement-to-track paths along a batch, the assignment of these costs-to-track can be performed via a variety of methods. In [21], an efficient near-optimal algorithm is developed using a relaxation algorithm based on Lagrangian multipliers, with a modified Auction algorithm [29]. Other efficient methods are also currently available for solving the objective function for global constraint optimization problems [2]. Some of the other solutions were mentioned in Chapter I in the review of batch-oriented MTT algorithms. There is also a variety of global optimization solvers commercially available or as open source software packages. As part of this study, an open source package called LP_SOLVE [2], [3] is used to perform the linear global constraint optimization. LP_SOLVE utilizes a mixed integer linear programming relaxation technique via the primal-dual method. Linear programming (LP) was originally developed by Dantzig [30] in 1947 and was initially used by the U.S. Air Force. Currently, LP is used in a variety of fields to solve the linear global constraint optimization problem. LP sets up the optimization formulation in the form of $\mathbf{Ax} = \mathbf{B}$, where $\mathbf{A}$ contains the associated N-tuple costs c, $\mathbf{x}$ contains the binary decision variable ρ, and $\mathbf{B}$ contains the associated constraint terms as outlined in this study. A detailed description of LP is beyond the scope of this thesis, but the reader is referred to several texts [30], [31], [32] on this well-researched and much-applied method. The ND assignment operation is performed in the LP_Assign module of Figure 1, and objective formulation is outlined in the next section.

5. ND Assignment

The ND assignment refers to a batch length of $(N - 1)$ scans where the first dimension represents the prior state estimates and covariance of a given track. For example, a 16D assignment, or a batch length of 15 scans, is processed for each target. The ND assignment formulation follows Equations (25) and (26).

The objective function is given by

$$\min_{\rho_{i_1 i_2 \cdots i_n}} \sum_{i_1=u_1}^{m_1} \sum_{i_2=u_2}^{m_2} \cdots \sum_{i_n=u_n}^{m_n} c_{i_1 i_2 \cdots i_n} \rho_{i_1 i_2 \cdots i_n} , \tag{25}$$

where c is the N-tuple Kalman filter negative log likelihood, subject to the set of constraints defined as

$$\begin{aligned} &\sum_{i_2=u_2}^{m_2} \sum_{i_3=u_3}^{m_3} \cdots \sum_{i_n=u_n}^{m_n} \rho_{i_1 i_2 \cdots i_n} = 1 \quad \text{for } i_1 = 1, 2, \cdots m_1 \\ &\sum_{i_1=u_1}^{m_1} \sum_{i_3=u_3}^{m_3} \cdots \sum_{i_n=u_n}^{m_n} \rho_{i_1 i_2 \cdots i_n} = 1 \quad \text{for } i_2 = 1, 2, \cdots m_2 \\ &\vdots \qquad\qquad\qquad\qquad \vdots \\ &\sum_{i_1=u_1}^{m_1} \sum_{i_2=u_2}^{m_2} \cdots \sum_{i_{n-1}=u_n}^{m_n-1} \rho_{i_1 i_2 \cdots i_n} = 1 \quad \text{for } i_n = 1, 2, \cdots m_n , \end{aligned} \tag{26}$$

where m_n is the number of measurements for scan n from 1 to $(N - 1)$. The measurement index i ranges from u_n to m_n, where $u_n = 0$, and n is the scan index. Note that ($u_n = 0$) is the missing measurement. The values of $c_{i_1 i_2 \cdots i_n}$ are the N-tuple costs. These costs are the cumulative Kalman filter negative log likelihoods for the batch, Λ, as obtained in the GenCost function. The costs are minimized in Equation (25) subject to the constraints in Equation (26). The constraints impose the requirement that at most one measurement is associated with a track per scan in a batch and multiple tracks can be associated with the missing measurement at the same scan. In other words, each measurement-to-track assignment is exclusive per scan in the batch. It is noted in [27] that the actual measurement-to-track assignment is a very efficient process and only takes 5% of the computational resources for the algorithm. Once the measurement-to-track assignments are obtained, the next step, as shown in Figure 1, is to update the track's state estimate via a standard Kalman filter as described in the next section.

6. Kalman Filter Update

Once the measurement-to-track assignment is performed, a Kalman filter is used to update the state matrix for the next scan t_s. The filter process is performed via the Kalman filter state space Equations (5) through (13), calculated for each track at current scan t_s with measurement $z(t_1)$ obtained for the track via the (LP_Assign) assignment function for the first scan of the batch. Note that, in Equations (5) through (13), the scan index n represents the scan index in a batch, which is replaced in this step with s, the scan epoch index for the total number of scans.

Note that an assignment of a missing measurement only performs the Kalman time update, Equations (5) through (8). The filter is processed, and the cycle repeats for the next scan of data according to Figure 1.

D. ALGORITHM II: A FASTER, SUBOPTIMAL VERSION VIA 2D-ND MDA

1. Overview of Algorithm II

The primary goal of this research is to extend the batch length size of the tracker with the premise that it will aid in the long-duration target crossings problem. In Algorithm I, it is shown that by increasing the dimensional size, the number of Kalman filters to process the cost evaluations increases exponentially. In [27], it is noted that even though the assignment function of the MDA is the most important part of the algorithm, it accounts for only 5% of the computational resources. About 95% of the computational load is due to the generation of the cost evaluations, to form the objective function, given by Equation (25). In Algorithm I, the number of cost evaluations remained the optimal size with a full enumeration of all N-tuple pairwise measurement-to-track decisions along the batch were calculated. In [27], a fast MDA style algorithm is developed by using a clustering process to reduce the number of cost evaluations. This is a type of gating procedure along a batch based on a clustering approach. As part of this research, in Algorithm II, a method is proposed to reduce the cost evaluations by gating the tracks and identifying single target and multiple target groupings. The tracking of these groups is performed with either a single scan (2D single-target) or multiscan (ND multitarget) multidimensional assignment. An overview of Algorithm II of MDA is shown in Figure 4. When multiple tracks are within a gate, the algorithm reduces the number of cost evaluations by employing a filtering of the measurements by a 2D LP assignment via a forward-looking filter along the batch. These filtered measurements are reprocessed by an ND LP assignment MDA. When tracks are outside the vicinity of any other tracks—basically, a single target track—the algorithm reverts to a 2D algorithm as outlined in Algorithm I, with *N* equal to 2.

The 2D assignment (single-target mode) is very efficient due to this linearity in the number of measurements to tracks. For example, if 400 scans were processed with two tracks and three measurements per scan, a total of 2400 cost evaluations or Kalman filters are computed for this scenario. In the 4D example given in the Algorithm I section of the thesis, with two tracks, three measurements per scan, with 400 scans to be processed, a total of 21,600 cost decisions or 31,200 Kalman filters need to be calculated for this example. In this case, the 4D version is 13 times more expensive than the 2D version with three measurements per scan. This savings is the rationale for using the 2D LP when in single target mode. When there is no contention among close by tracks, which is the single target case, a 2D assignment is a cost efficient tracker. It is also noted that the 2D assignment degrades in performance as the clutter level increases. In simulation, the 2D assignment also tended to exhibit the track repulsion behavior for long-duration crossing targets more than versions with longer batch lengths.

It is emphasized that in the multiscan, multitarget (ND) mode in Algorithm II, the Kalman filter processes the entire batch and then skips forward to process the next batch of data out of the gate. This further increases the speed of the algorithm over the processing from scan to scan described for Algorithm I. This part of the processing is described in Section D.5 for Algorithm II.

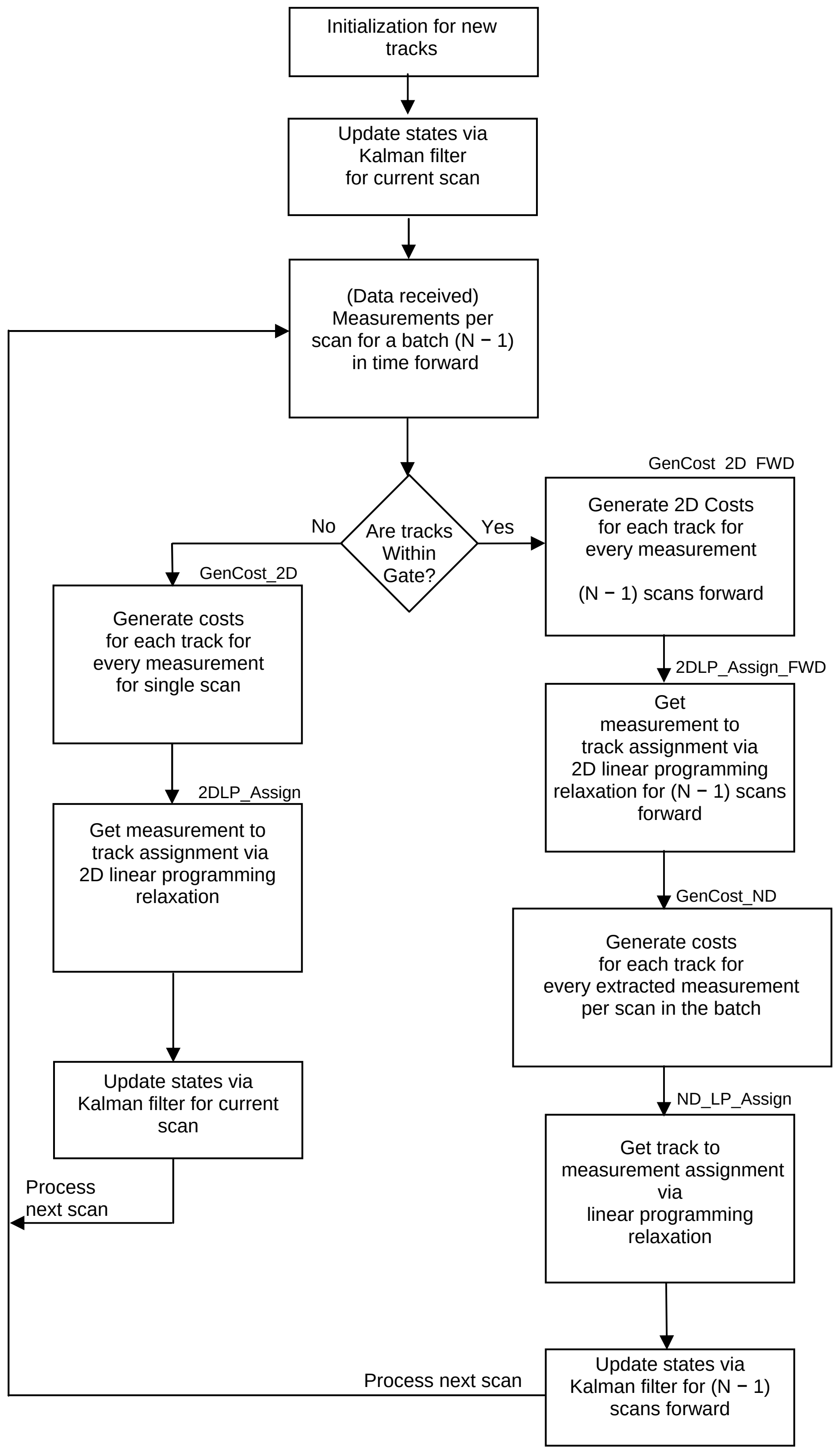
Initialization for new tracks
Update states via Kalman filter for current scan
(Data received) Measurements per scan for a batch (N − 1) in time forward
Are tracks Within Gate?
No
Yes
GenCost_2D
Generate costs for each track for every measurement for single scan
2DLP_Assign
Get measurement to track assignment via 2D linear programming relaxation
Update states via Kalman filter for current scan
Process next scan
GenCost 2D FWD
Generate 2D Costs for each track for every measurement (N − 1) scans forward
2DLP_Assign_FWD
Get measurement to track assignment via 2D linear programming relaxation for (N − 1) scans forward
GenCost_ND
Generate costs for each track for every extracted measurement per scan in the batch
ND_LP_Assign
Get track to measurement assignment via linear programming relaxation
Update states via Kalman filter for (N − 1) scans forward
Process next scan

Figure 4. Overview of Algorithm II: 2D-ND LP MDA tracking algorithm.

2. Track Initialization

The track initialization functionality is the same as described in Algorithm I. This function is performed at the first scan, and states are projected forward for the next scan.

3. Track Gating

When multiple tracks are within a gate, there are two possible outcomes from Algorithm II. When the tracks are in gate proximity, the 2D-ND LP tracker (multitarget tracking) is employed to process the multiple tracks. When the tracks are separated, outside any tracker gates, the 2D LP tracker (single-target tracking) is performed. The gating procedure can be performed either spatially or temporally. For example, a spatial gating can be performed between multiple tracks using the Kalman filter negative log likelihoods described in Algorithm I with an appropriately set gating threshold. In this research, a time-dependent gating procedure is used. Because the goal of this study is the long-duration crossing, a time window is identified in which the 2D-ND LP tracker is processed. The 2D LP tracker is used for all scans before and after the window, which begins with time scan t_{n_start} and ends with time scan t_{n_end}.

The pseudo code for the 2D LP tracker (single-target case) is as follows:

```
if $t_n < t_{n\_start}$ or $t_n > t_{n\_end}$ .
```

The pseudo code for the 2D-ND LP tracker (multitarget case) is as follows:

```
else ($t_n \geq t_{n\_start}$ and $t_n \leq t_{n\_end}$).
```

A diagram of the time-dependent track gating is given in Figure 5. This methodology is used for consistency in the simulated results given in Chapter III. A spatial gate can be easily interchanged as part of the gating procedure within this paradigm. Tracks that are about to cross can also be identified in time via extrapolation using a linear least squares fit along the track or another procedure. This research uses fixed time scans for switching between the two paths, as shown in the flow diagram in Figure 4.

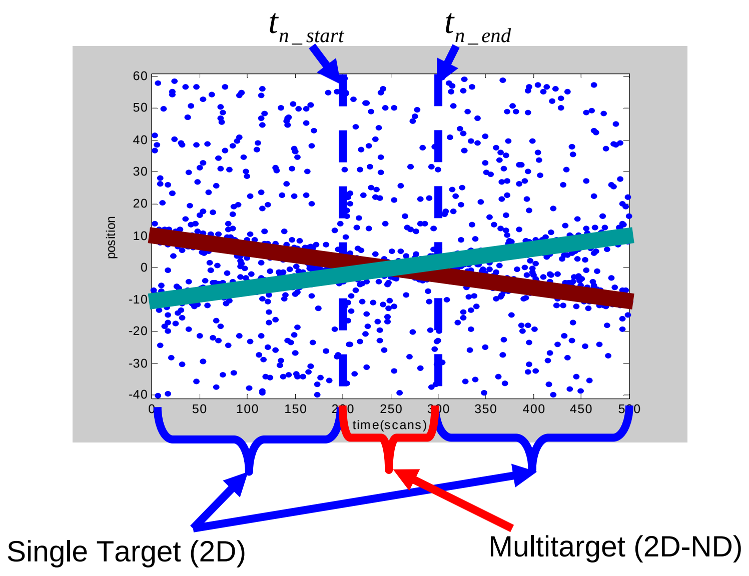

Figure 5. Track gate switching is performed at time scans t_{n_start} and t_{n_end}. The 2D LP tracker (single-target, single-scan) is used prior to t_{n_start} and after t_{n_end}. The 2D-ND LP tracker (multitarget, batch) is used between time scans t_{n_start} and t_{n_end}.

4. 2D LP Assignment Single Scan

When tracks are outside of a gate, a 2D LP is used to update the state estimates and the next scan is processed. The costs are calculated as in Algorithm I outlined in the Cost Generation section. The 2D assignment is performed as outlined in the ND assignment section of Algorithm I with N equal to 2. The modules GenCost_2D and 2DLP_Assign shown in Figure 4 are special situations of the generic GenCost and LP_Assign modules as described in Algorithm I. Note that in Tables 2 and 5 that the number of cost evaluations and/or Kalman filters to process is linear with the number of measurements M (including missing measurement) in the scan for the 2D assignment.

5. 2D-ND LP Assignment Multiscan

When tracks are inside of a gate, a 2D-ND LP is processed. The modules shown in Figure 4 labeled GenCost_2D_FWD, 2DLP_Assign_FWD, GenCost_ND, and ND_LP_Assign account for the main processing chain of this algorithm.

In the GenCost_2D_FWD, and 2DLP_Assign_FWD, a 2D LP filter is run $(N-1)$ scans forward in time. This process is similar to the section in Algorithm II for the 2D LP Assignment for a single scan. The measurements that were assigned to the track are saved to be reprocessed by ND LP assignment. During the assignment process, it is noted that a track received an actual measurement or a missing measurement. By following this step, the number of measurements m_n processed by the ND Assignment becomes equal to or less than the number of tracks L. In most circumstances, the number of tracks is less than the number of measurements. Note that using the 2D assignment as a filter, there are certain advantages especially when processing in low clutter. First, if all the tracks are assigned an actual measurement on a particular scan in the batch, this is duly noted and the missing measurement need not be processed for that scan. Second, if none of the tracks is assigned a real measurement, then only the missing measurement needs to be processed for the scan. Third, as the clutter level λ increases, the Kalman filter calculation costs are absorbed by the 2D assignment, which is linear with the number of measurements. The filtered data sent to the ND Assignment is always less than or equal to the number of tracks no matter how large the clutter level. This property has benefits and disadvantages. In dense clutter, this part of the algorithm is very fast but performance also degrades. Note that the 2D assignment process discussed in this section is only used as a filtering process. The state estimates derived from this process are not used at the end of each batch. The 2D assignment filters the batch of data and forms a new filtered batch of data to be reprocessed by the ND assignment.

In the GenCost_ND, and ND_LP_Assign sections, these modules are similar to the GenCost and LP_Assign shown in Figure 1 of Algorithm I. The difference is in Algorithm I, the missing measurement is accounted on every scan of the batch. In the ND assignment in Algorithm II, the 2D assignment provides an index if a missing measurement or a real measurement is processed for a particular scan. Therefore, the missing measurement need not be processed for every scan in the batch. By filtering with the 2D assignment, the number of 2D LP filtered measurements l_n, to process with the ND assignment is less than or equal to the number of tracks L. The ND assignment formulation for Algorithm II has the same set of Equationss (25) and (26) as in Algorithm I.

The objective function in Equation (25) is minimized subject to the constraints in Equation (26). The binary value u_n is the measurement index value filtered by the 2D LP forward filter. If u_n equals 0, a missing measurement is processed; otherwise, the index is 1, representing the first real measurement. The value m_n equal to l_n, is the number of 2D LP filtered measurements for scan n from 1 to $(N - 1)$. The measurement index i ranges from 0—the missing measurement—to ($m_n = l_n$), where l_n is the number of measurements filtered by the 2D forward filter per scan, and n is the scan index.

The maximum number of costs calculated per track for the 2D-ND LP MDA for a fixed number of measurements per scan in a batch is given by Equation (27):

$$\text{Number of costs } (c) = M(N-1) + L^{N-1}, \tag{27}$$

where M is the number of measurements per scan including the missing measurement, $(N - 1)$ is the batch length, and L is the number of tracks. The total number of costs is given by Equation (28):

$$\text{Total number of costs} = (M(N-1) + L^{N-1})L\,. \tag{28}$$

The total number of Kalman filters needed to process a single track in a batch is given by Equation (29):

$$\text{Number of KF in reduced set} = M(N-1) + \sum_{n=1}^{N-1} L^n\,,$$

or (29)

$$\text{Number of KF in reduced set} = M(N-1) + \frac{L^N - L}{L-1}\,,$$

where L > 1 in the generation function.

The generic equations used to calculate the number of costs and the number of Kalman filters in the reduced set for any number of measurements in a batch for a single track are given as Equations (30) and (31), respectively.

$$\text{Number of costs } (c) = \sum_{n=1}^{N-1} m_n + \prod_{n=1}^{N-1} l_n \,, \tag{30}$$

$$\text{Number of reduced KF} = \sum_{n=1}^{N-1} m_n + \sum_{k=2}^{N} \prod_{n=1}^{k-1} l_n \,, \tag{31}$$

where l_n is the number of 2D LP extracted measurements for scan n from 1 to $(k-1)$ and k is the dimensional assignment from 2 to *N*. Note that $l_n \le L$ where L is the number of tracks for any scan n.

6. Kalman Filter Update Multiscan

A Kalman filter update is performed for every scan in the batch based on the measurements obtained from the ND assignment. Note that this is a different operation than Algorithm I, where a sliding batch is employed and only the first value in the batch is used to update the Kalman filter. In Algorithm II, the Kalman filter processes the entire batch, and skips forward to process the next batch of data until the track is out of the gate. The Kalman filter follows Equations (5) through (13) for scan index t_s.

7. Example

As an example, consider a case with 500 scans, two tracks ($L = 2$), and three measurements including the missing measurement (M = 3) that uses a 6D assignment (*N* = 6). The tracks are separated during the first 200 scans, crossing during the next 100 scans, and separated again during the last 200 scans. A diagram of this example is shown in Figure 6. Using Algorithm I, the total number of Kalman filters (KF) for the separated tracks, i.e., for 400 scans, in this case is calculated as Num_KF_*M*_*N* $*$ *L* $*$ Num_scan, where Num_KF_*M*_*N* is the number of KFs from Table 5 for given values of *M* and *N*, and Num_scan is 2 $*$ 200 scans, or 400. Thus, the total number of KFs for the separated tracks is 363 $*$ 2 $*$ 400, or 290,400 KF. From Table 2, the total number of cost evaluations is calculated as cost_*M*_*N*, where cost_*M*_*N* is the number of costs from Table 2 for given values of *M* and *N*. Thus, cost_*M*_*N* = *c* $*$ *L* $*$ Num_scan = 243 $*$ 2 $*$ 400, or 194,400 cost evaluations.

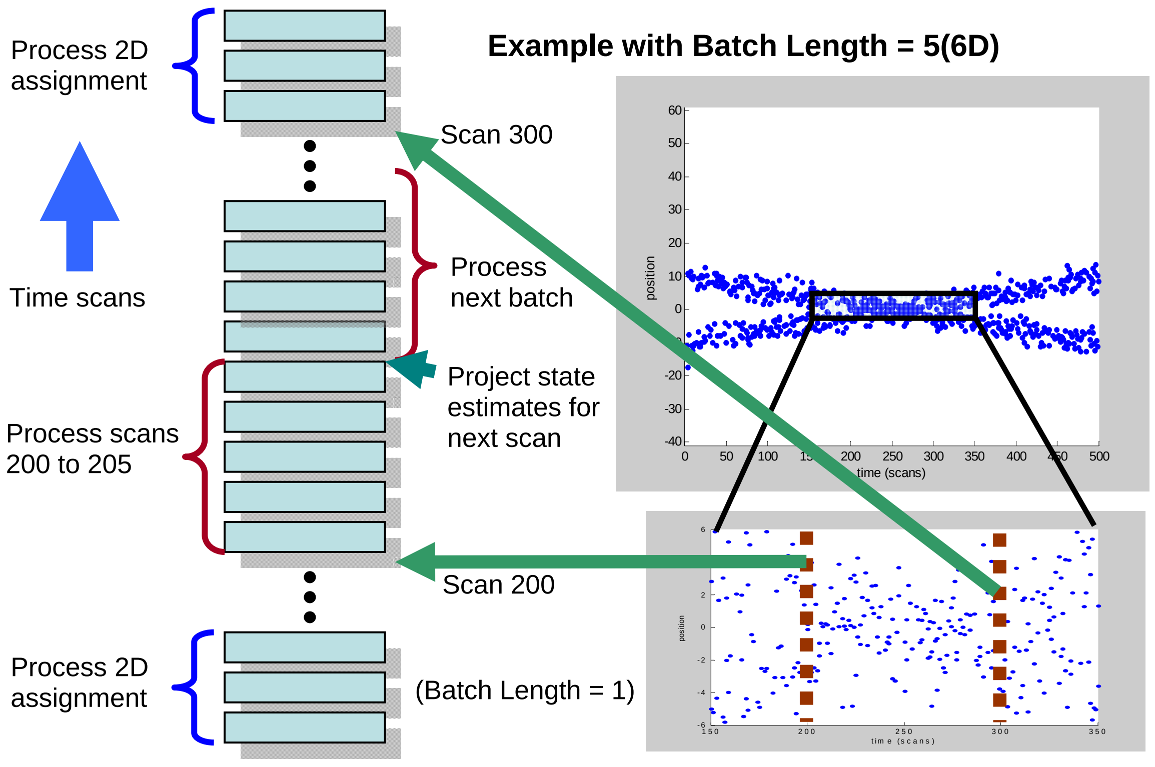

Figure 6. Example of 6D assignment for two-target case with 500 scans and target crossings for 100 scans.

Using Algorithm II for the single-target section, where tracks are outside the gate of Figure 4 (i.e., separated), the total number of KFs for the separated tracks is calculated as Num_KF_2D_M_N ∗ L ∗ Num_scan, where Num_KF_2D_M_N is the number of KFs from Table 5 for given values of M and N, and Num_scan is 400. Thus, the total number of KFs for the separated tracks using Algorithm II is 3 ∗ 2 ∗ 400, or 2400 KFs. From Table 2, the total number of cost evaluations is calculated as cost_2D_M_N, where cost_2D_M_N is the number of costs from Table 2 for given values of M and N. Thus, cost_2D_M_N = c ∗ L ∗ Num_scan = is 3 ∗ 2 ∗ 400, or 2400 cost evaluations. Thus, based on the ratio of the total number of KFs, Algorithm II is 121 times faster than Algorithm I for this case.

Using Algorithm I for the multitarget case, the tracks are within a gate between scans 200 and 300, i.e., Num_scan = 100. From Table 5, the number of Kalman filters to process for Algorithm I is Num_KF_M_N ∗ L ∗ Num_scan = 363 ∗ 2 ∗ 100, or 72,600

KF. From Table 2, the total number of cost evaluations cost_*M*_*N* is calculated as $c * L *$ Num_scan = 243 ∗ 2 ∗ 100, or 48,600 cost evaluations.

Using Algorithm II for the multitarget case requires two steps to calculate the number of Kalman filters. The 2D LP function is used for forward filtering, and then the ND LP function is used to calculate the cost based on the 2D LP filtered output. Thus, the total number of KFs for the multitarget case = (Num_KF_2D_*M*_*N* ∗ *L* * Num_scan) + (Num_KF_6D_*L*_*N* ∗ *L* ∗ Num_scan/($N - 1$)) = (3 ∗ 2 ∗ 100) + (62 ∗ 2 ∗ 20), or 3080 KFs. Thus, based on the ratio of the total number of KFs, Algorithm II is about 23 times faster than Algorithm I for the multitarget case.

From Table 2, the total number of cost evaluations *c* is calculated as (cost_2D_*M*_*N* ∗ *L* ∗ Num_scan) + (cost_6D_*L*_*N* ∗ *L* ∗ Num_scan/($N - 1$)) = (3 ∗ 2 ∗ 100) + (32 ∗ 2 ∗ 20) = 1880 cost evaluations. The number of cost evaluations is reduced by more than 25 times by using Algorithm II instead of Algorithm I. It is important to note that much of the savings is attributed to the elimination of the sliding batch in Algorithm II and the 2D LP data filtering that reduces the number of measurements per scan from *M* to the number of tracks *L*. The 2D LP for the single-target section of the algorithm provides most of the computational savings. The comparison of costs and number of reduced Kalman filters between Algorithm I and Algorithm II is provided in Table 7.

	Algorithm I: ND LP	Algorithm II: 2D-ND LP
No. of cost evaluations c	$\prod_{n=1}^{N-1} m_n$	$\sum_{n=1}^{N-1} m_n + \prod_{n=1}^{N-1} l_n$
No. of reduced KF	$\sum_{k=2}^{N} \prod_{n=1}^{k-1} m_n$	$\sum_{n=1}^{N-1} m_n + \sum_{k=2}^{N} \prod_{n=1}^{k-1} l_n$

Table 7. Comparison of Algorithm I and II functions for calculating the number of cost evaluations and the number of Kalman filters in a reduced set for a single track and a batch length ($N - 1$), where *m* is the varying number of measurements for scan *n* from 1 to ($k - 1$), and *k* is the dimensional assignment from 2 to *N*. The value l_n is the number of 2D LP extracted measurements for scan *n* from 1 to ($k - 1$), and *k* is the dimensional assignment from 2 to *N*. Note that, for any scan *n*, $l_n \leq L$ where *L* is the number of tracks.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SIMULATION AND RESULTS

The following simulation is based on a study conducted by Willet et al. [1] for various MHT batch-style algorithms for a passive sonar line array. The results from [1] are not reproduced as part of this thesis. Figure 7 shows a sample case with two targets crossing in a cluttered environment where the crossing occurs over 100 scans. The solid green lines represent the true target trajectories. The zoom view on the right shows the difficulty a tracker would have in interpreting the data during the long-duration crossing of 100 scans. The goal of this study is to extend the batch length, using MDA or another method, to beyond the length of the crossing. The objective for the tracking algorithm is to maintain track on all targets through the crossing. Because of the exponential nature of MDA and the current state of computing resources, it is impractical to develop a tracker with acceptable runtime performance for batch lengths of 100 scans or more without major pruning of the cost evaluations or the use of other suboptimal techniques. Algorithm II provides a more efficient method for extending the batch length while still adhering to the basic concepts of MDA.

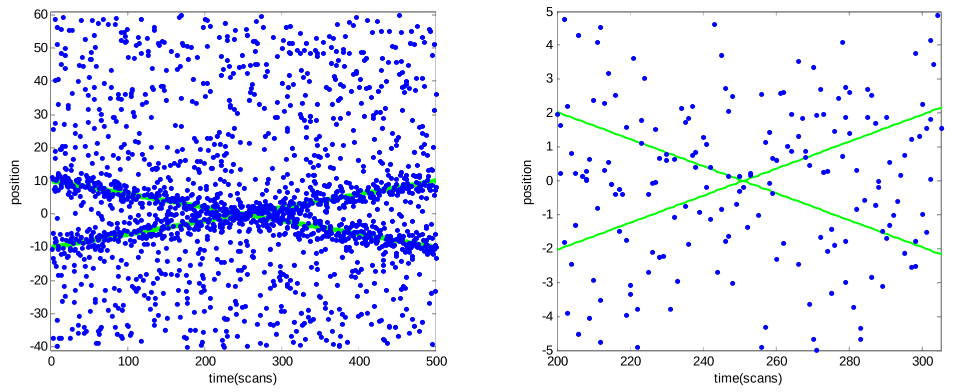

Figure 7. Sample simulated case with two targets in linear straight-line motion through a 100-scan crossing in a cluttered environment, $\lambda = 0.01$ (left). In a zoom view in the area of the target crossing, or approximately 100 time scans (right), the solid green lines represent the true target trajectories.

In all of the simulations, the P_D = 70%, and σ_r = 2, where P_D is the probability of detection, and σ_r is the measurement error standard deviation. Thus, when σ_r = 2, the targets are within $2\sigma_r$ of one another during the 100 scans of crossing. The Kalman filter's initial process noise variance $\sigma_{f_0}^2$ equals 0.0625. In these simulations, the value for σ_q, the process noise standard deviation for the Kalman filter, is set to 0.0005, which corresponds to a final position standard deviation of $T\sigma_q\left(n(n+1)(2n+1)/6\right)^{1/2}$ = 3.23. This value is important because it determines the stiffness of the track and/or the randomness of the simulated measurements as input to the tracker. λ = 0.0, 0.01, 0.02, and 0.05 are the clutter density levels for the trials. For instance, a clutter density of 0.01 corresponds to a single clutter measurement in a linear space of a 100 points. The negative Kalman filter likelihood function's clutter density parameter λ_e in Equation (14) is set at 0.01 for all cases. There were 100 trials conducted for each value of the clutter density level λ. Each trial consisted of 500 scans of data with two crossing targets. The tracks were initiated in all scenarios at 10 and −10 with zero velocity. Figure 8 shows sample clutter levels and track crossing scenarios used as input and output for the test results. The results from the 100 trials actually included five track crossing scenarios—when all tracks are tracking (*both tracking*), both tracks bounce (*switched*), the tracks coalesce (*coalesced*), only a single track is tracking (*one only*), and none of the tracks are tracking the target (*neither*). The coalesced outcome seldom occurred for the MDA algorithm—only for the high clutter case. This reflects the constraints imposed as part of the discrete optimization problem. The thin green and blue lines represent the truth of the target paths. The thicker solid green and blue lines represent the tracker outputs.

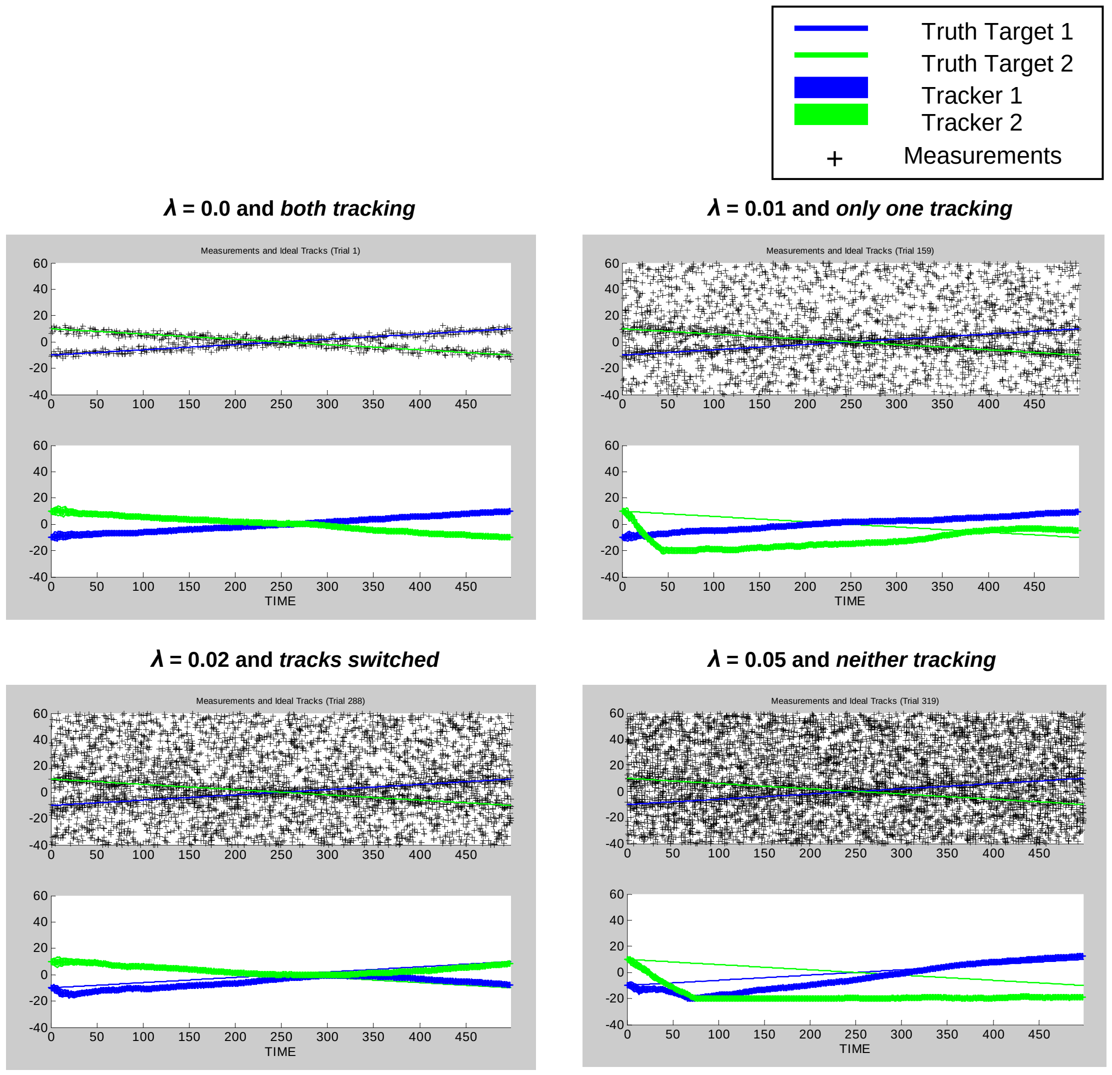

Figure 8. Sample clutter levels and track crossing scenarios used as input and output to obtain test results: zero clutter scenario with both tracks on target (top left), a clutter level of 0.01 and only one track on target (top right), a clutter level of 0.02 and both tracks bounce (bottom left), and a clutter level of 0.05 and neither track is on target (bottom right).

Figure 9 shows the MDA results of using Algorithm I, a typical sliding batch implementation of the MDA style tracker, with batch lengths ($N - 1$) of 1, 2, and 3 (corresponding to 2D, 3D, and 4D implementations of MDA). Past the 4D mark, the number of calculations to form the cost matrix is large (see Table 5). For 4D or greater, it is not possible to obtain the simulation runs in a reasonable length of time. The results show that with no clutter ($\lambda = 0.0$), the 2D assignment tracks both targets in more than

50% of the cases. The exact value is 56%, with 95% confidence intervals between 46.27% and 65.73%, based on Bernoulli trials. For batch lengths of 2 and 3, Algorithm I tracks correctly more than 70% of the time in no clutter. For a batch length of 2, 72% of the crossings are tracked correctly, with 95% confidence intervals between 63.2% and 80.8%. For a batch length of 3, the correct crossings increase to 75%, with 95% confidence intervals between 66.51% and 83.49%. When $\lambda = 0.01$, the correct tracking performance degrades to less than 30% of the time for all batch lengths. Table 8 gives the percentage of correct crossings (i.e., both targets tracked through the crossing) in the Algorithm I results, with 95% confidence intervals, for batch lengths of 1, 2, and 3 with $\lambda = 0.0$, 0.01, 0.02, and 0.05 for each batch length. Increasing the batch length from 2 to 3 does improve the performance slightly for $\lambda = 0.01$. As the clutter level is increased, the algorithm performance decreases as expected.

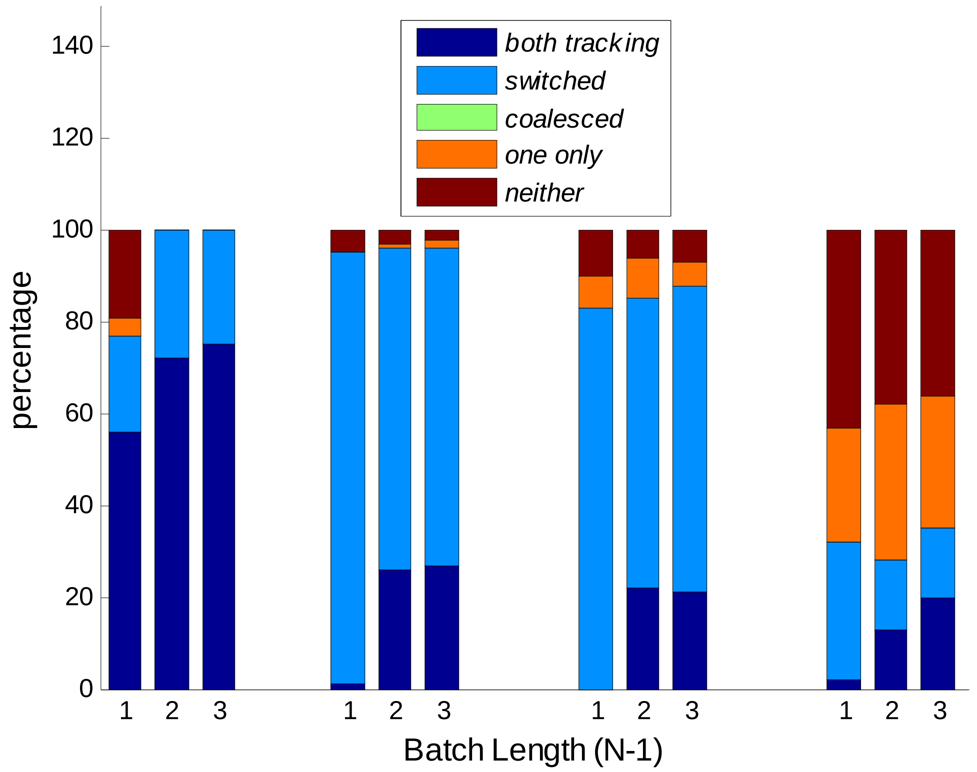

Figure 9. Results from Algorithm I: ND LP MDA version for straight-line model, $\sigma_q = 0.0005$. For each group of bars, the three numbers at the bottom (1, 2, 3) represent the batch length ($N - 1$), with 100 trials per bar. The groups, left to right, represent clutter density parameter λ at 0.0, 0.01, 0.02, 0.05.

Batch Length	λ	Correct Crossings (%)	95% Lower Bound	95% Upper Bound
1	0	56	46.27	65.73
1	0.01	1	0	2.95
1	0.02	0	0	0
1	0.05	2	0	4.74
2	0	72	63.2	80.8
2	0.01	26	17.4	34.6
2	0.02	22	13.88	30.12
2	0.05	13	6.41	19.59
3	0	75	66.51	83.49
3	0.01	27	18.3	35.7
3	0.02	21	13.02	28.98
3	0.05	20	12.16	27.84

Table 8. Algorithm I correct track crossings (both targets are being tracked) in 100 trials if σ_q = 0.0005, with 95% confidence intervals. Batch lengths are 1, 2, and 3 for λ = 0.0, 0.01, 0.02, and 0.05.

Because of the excessive time required to process batch lengths of a higher order with Algorithm I, Algorithm II was developed to speed up the processing time by combining the 2D and ND versions of the MDA algorithm. The 2D LP tracker is used until the tracks are about to cross. In all cases, the crossing occurs at scan 200. The 2D LP tracker is again used when the tracks exit the crossing at scan 305. As the track crossing occurs, a 2D-ND LP tracker is used with various batch lengths. The results from these trials are shown in Figure 10. For the value of σ_q = 0.0005, the algorithm performs very well for batch lengths of 2 through 10, with 99% tracking of both targets in moderate clutter (λ = 0.01).

Table 9 gives the percentage of correct crossings (both targets tracked through the crossing) in the Algorithm II results, with 95% confidence intervals, where σ_q = 0.0005, for batch lengths of 1, 2, 3, 5, and 10 with λ = 0.0, 0.01, 0.02, and 0.05 for each batch length. The performance degrades with increasing clutter as expected.

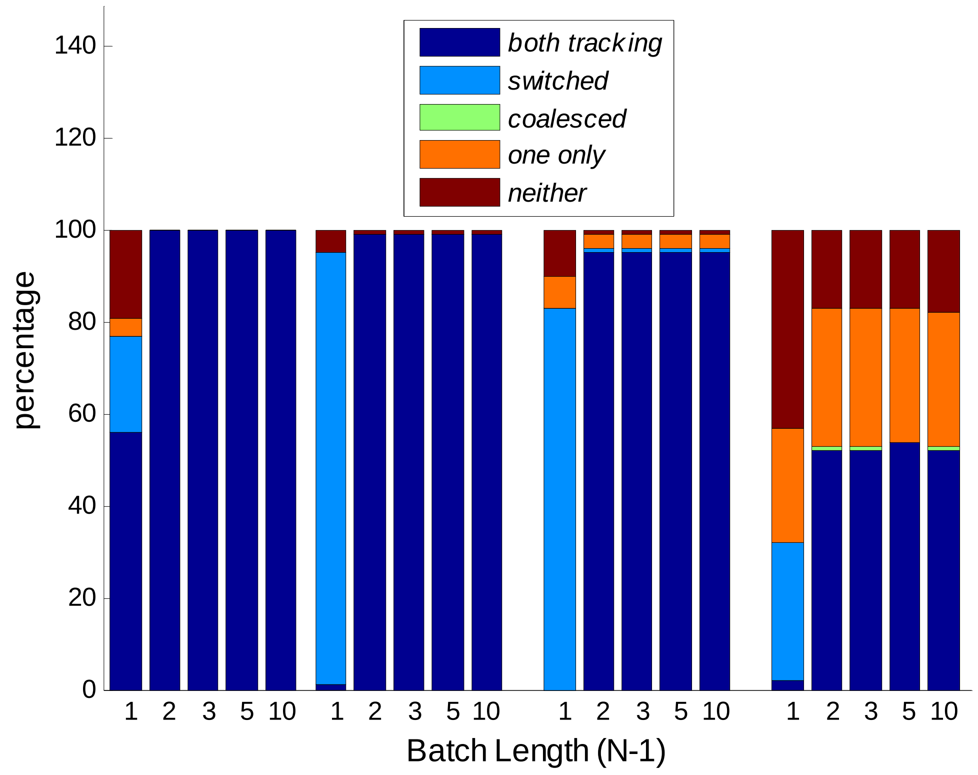

Figure 10. Results from Algorithm II: 2D-ND MDA fast version, $\sigma_q = 0.0005$. For each group of bars, the five numbers at the bottom (1, 2, 3, 5, 10) represent the batch length ($N - 1$), with 100 trials per bar. The groups, left to right, represent the clutter density parameter λ at 0.0, 0.01, 0.02, 0.05.

Batch Length	λ	Correct Crossings (%)	95% Lower Bound	95% Upper Bound
1	0	56	46.27	65.73
1	0.01	1	0	2.95
1	0.02	0	0	0
1	0.05	2	0	4.74
2	0	100	100	100
2	0.01	99	97.05	100
2	0.02	95	90.73	99.27
2	0.05	52	42.21	61.79
3	0	100	100	100
3	0.01	99	97.05	100
3	0.02	95	90.73	99.27
3	0.05	52	42.21	61.79
5	0	100	100	100
5	0.01	99	97.05	100
5	0.02	95	90.73	99.27
5	0.05	54	44.23	63.77
10	0	100	100	100
10	0.01	99	97.05	100
10	0.02	95	90.73	99.27
10	0.05	52	42.21	61.79

Table 9. Algorithm II correct track crossings (both targets are being tracked) in 100 trials if $\sigma_q = 0.0005$, with 95% confidence intervals. Batch lengths are 1, 2, 3, 5, and 10 for $\lambda = 0.0$, 0.01, 0.02, and 0.05.

Because the Algorithm II results were good with σ_q set to 0.0005, σ_q was increased to 0.002 for the four clutter levels used in the prior trials. These results are shown in Figure 11 for batch lengths of 1, 2, 3, 5, 10, and 15. The results indicate that the Algorithm II tracking performance is better than 90% in terms of correct crossings for batch lengths greater than 1 with no clutter.

Table 10 gives the percentage of correct crossings (both targets tracked through the crossing) in the Algorithm II results, with 95% confidence intervals, where $\sigma_q = 0.002$, for batch lengths of 1, 2, 3, 5, 10, and 15 with $\lambda = 0.0$, 0.01, 0.02, and 0.05 for each batch length. As the clutter level increases, the tracking performance decreases proportionately for batch lengths greater than 1. At each level of clutter above $\lambda = 0.0$, the tracking performance increases as the batch length increases. With $\sigma_q = 0.002$, the trend in the results for clutter levels λ of 0.01, 0.02, and 0.05 indicates that increasing the batch length leads to a higher probability of maintaining track for long-duration crossings of 100 scans.

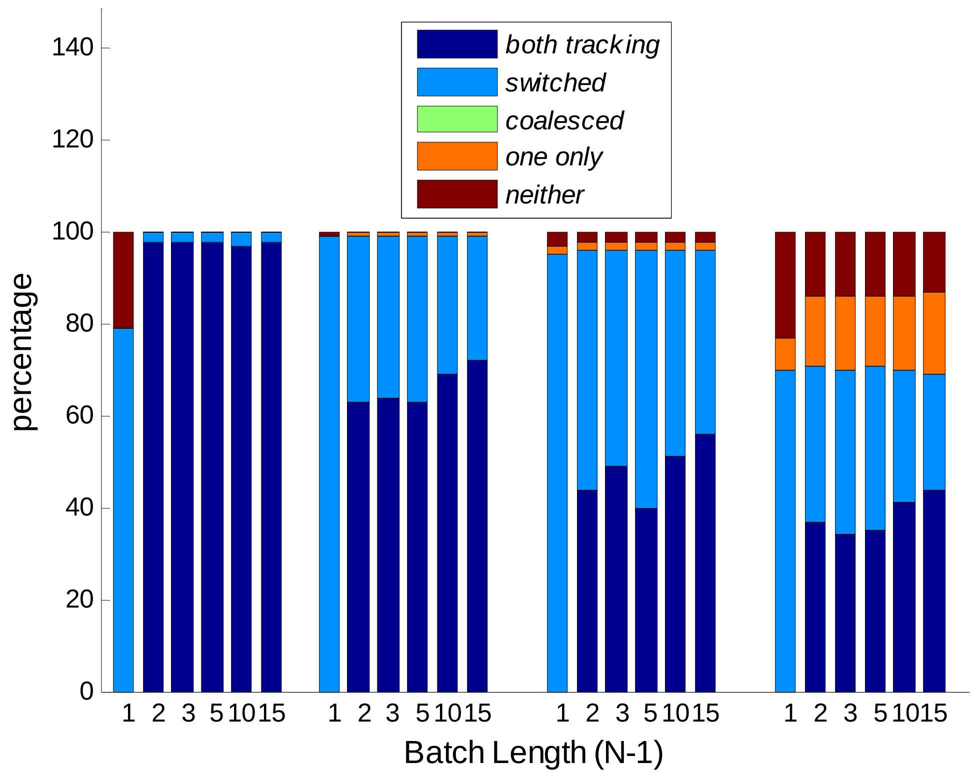

Figure 11. Results from Algorithm II: 2D-ND MDA, $\sigma_q = 0.002$. For each group of bars, the six numbers at the bottom (1, 2, 3, 5, 10, 15) represent the batch length $(N - 1)$, with 100 trials per bar. The groups, left to right, represent clutter density parameter λ at 0.0, 0.01, 0.02, 0.05.

Batch Length	λ	Correct Crossings (%)	95% Lower Bound	95% Upper Bound
1	0	0	0	0
1	0.01	0	0	0
1	0.02	0	0	0
1	0.05	0	0	0
2	0	98	95.26	100
2	0.01	63	53.54	72.46
2	0.02	44	34.27	53.73
2	0.05	37	27.54	46.46
3	0	98	95.26	100
3	0.01	64	54.59	73.41
3	0.02	49	39.2	58.8
3	0.05	34	24.72	43.28
5	0	98	95.26	100
5	0.01	63	53.54	72.46
5	0.02	40	30.4	49.6
5	0.05	35	25.65	44.35
10	0	97	93.66	100
10	0.01	69	59.94	78.06
10	0.02	51	41.2	60.8
10	0.05	41	31.36	50.64
15	0	98	95.26	100
15	0.01	72	63.2	80.8
15	0.02	56	46.27	65.73
15	0.05	44	34.27	53.73

Table 10. Algorithm II correct track crossings (both targets are being tracked) in 100 trials if $\sigma_q = 0.002$, with 95% confidence intervals. Batch lengths are 1, 2, 3, 5, 10, and 15 for $\lambda = 0.0$, 0.01, 0.02, and 0.05.

The appendix presents the normalized estimation error squared (NEES) and the root-mean-square error (RMSE) [25, pp. 234–235] for the average when both targets are being tracked correctly through the long-duration crossing for each of the test cases as a function of batch length and clutter density λ. The 95% confidence bounds for the two dimensions (position and velocity) are plotted and labeled on all of the NEES plots. In all cases, the 95% chi square distribution lower bound and upper bound are 0.0506 and 7.3778, respectively.

Notice that the tracker performance degrades with increasing clutter density λ in both the NEES plots and RMSE curves for both Algorithms I and II and also in the case where σ_q is increased. This effect is due to the nature of the MDA and other non-Bayesian association algorithms, such as nearest neighbor, strongest neighbor, and track split approaches [13, pp. 119–141], where a Kalman filter is used to estimate the state vector. A Kalman filter's estimate does not update its covariance update to measure its uncertainty as to whether the associated assignment originated from the target or was associated to clutter.

In the MDA batch approach, all measurement-to-track assignments are considered along the batch, but based on the constraints imposed, the maximum likelihood set of measurements is used to update the Kalman filters associated to the prior tracks. In the simulation being studied here, two tracks are initiated, and because a standard Kalman filter is utilized for state estimation, any association errors due to increasing clutter will detrimentally affect the tracker performance. Longer batch lengths for Algorithm I should improve the performance as clutter increases because the measurement-to-track assignment is exhaustive along the batch and also a longer time frame would be available to choose the correct path from the hypothesis tree. In Algorithm II, even though greater batch lengths are more feasible, the approach is considered suboptimal because a 2D forward filter is used to prune the measurement set. Any errors in the 2D association process will influence the subsequent ND assignment process and affect the tracker's results.

In Bayesian association algorithms, such as PDA and MHT, an exhaustive set of probabilities are calculated for every measurement-to-track pair. In PDA-style algorithms [13, pp. 119–141], the measurement origin uncertainty is integrated as part of the tracking filter, and the covariance estimate is duly updated to properly measure the consistency of a track. In MHT style algorithms [11], [12], although a Kalman filter is used to estimate the state vector, the initialization of tracks is incorporated as part of the tracking function, and every possible hypothesis is calculated for all scans in time, including the possibility that a measurement originated from a new target. This is considered an optimal approach.

IV. CONCLUSIONS

This study investigated two versions of a batch-oriented, MDA tracking algorithm to examine target crossings on the order of 100 scans in duration as tracked by a passive sonar line array. Many of the computational load issues in the generation and evaluation of the costs were identified to form the objective function. Linear programming relaxation was used to solve the assignment problem. A suboptimal but faster version of the ND assignment was developed to observe the effect of increased batch length on tracking through crossings. Batch lengths of up to 15 scans, or 16D assignment, were developed and tested on data with various levels of clutter for the suboptimal version. The two algorithms were tested via 100-trial Monte Carlo simulations.

Algorithm I, the optimal version, was tested up to the 4D assignment case. The results showed track repulsion behavior dominating correct tracking through crossings, which was achieved in less than 30% of the trials; the tracks were switched more than 60% of the time in moderate clutter ($\lambda = 0.01$). With a batch length of two and $\lambda = 0.01$, tracks were switched 70% of the time, with 95% confidence intervals between 61.02% and 78.98%, based on Bernoulli trials. With a batch length of three and $\lambda = 0.01$, tracks were switched 69% of the time, with 95% confidence intervals between 59.94% and 78.06%. Table 8 gives the percentage of correct track crossings for each case.

Algorithm II—suboptimal but faster—exhibited significantly better performance: the percentage of correct track crossings exceeded 99% for batch lengths of two or more and $\lambda = 0.01$, using the same process noise standard deviation as for Algorithm I. Table 9 gives the percentages observed in each case and the confidence intervals. When the standard deviation was increased for the Algorithm II results, the trend showed the number of correct track crossings generally increasing as batch length increased. A definitive conclusion cannot be drawn on this point because the spread in the data, <10%, is not statistically significant. The true effect of batch length on correct crossings can only be verified by testing with batches much longer (i.e., $(N - 1) > 100$ scans) than the 15 scans used here.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

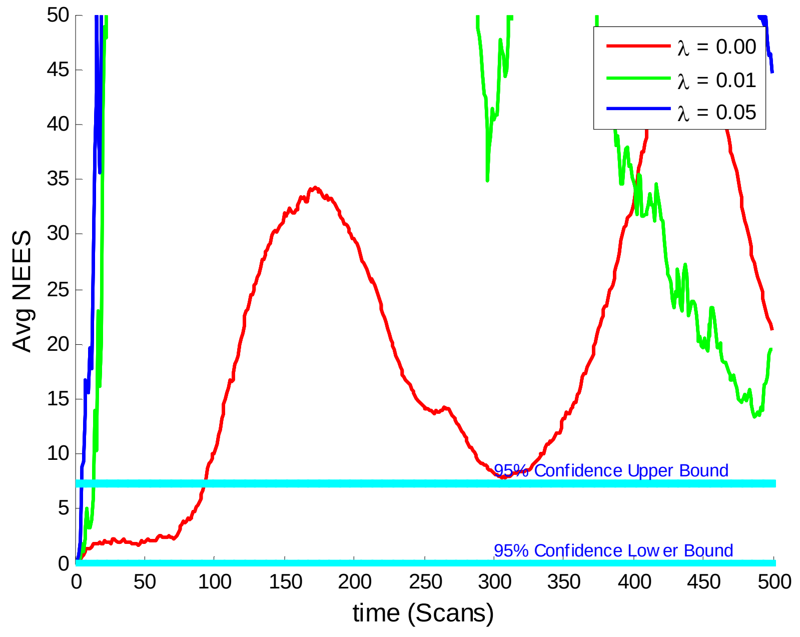

Figure 12. Average NEES for Algorithm I: 2D LP, $\sigma_q = 0.0005$.

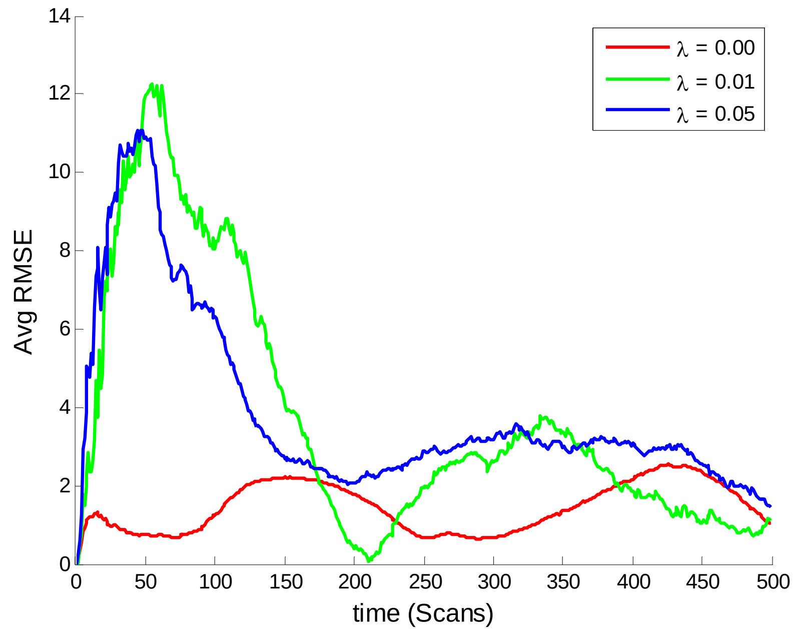

Figure 13. Average RMSE for Algorithm I: 2D LP, $\sigma_q = 0.0005$.

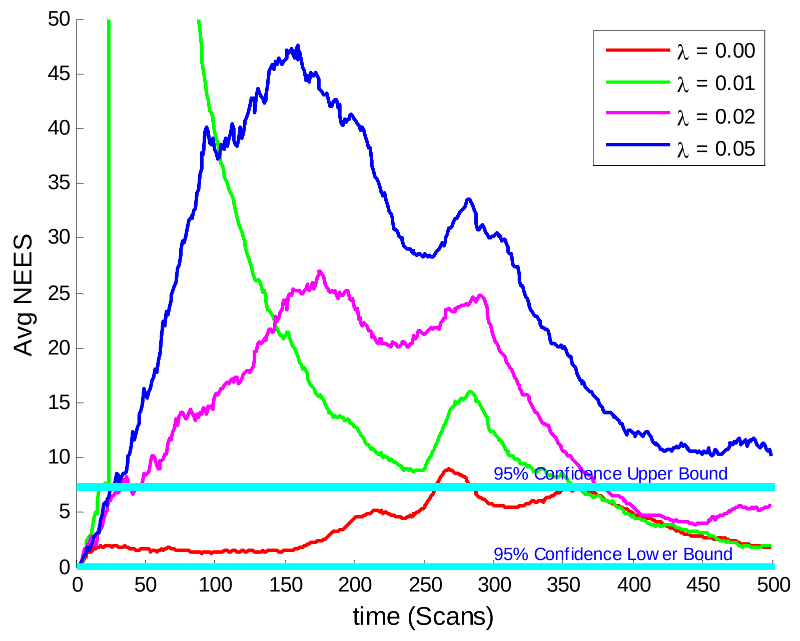

Figure 14. Average NEES for Algorithm I: 3D LP, $\sigma_q = 0.0005$.

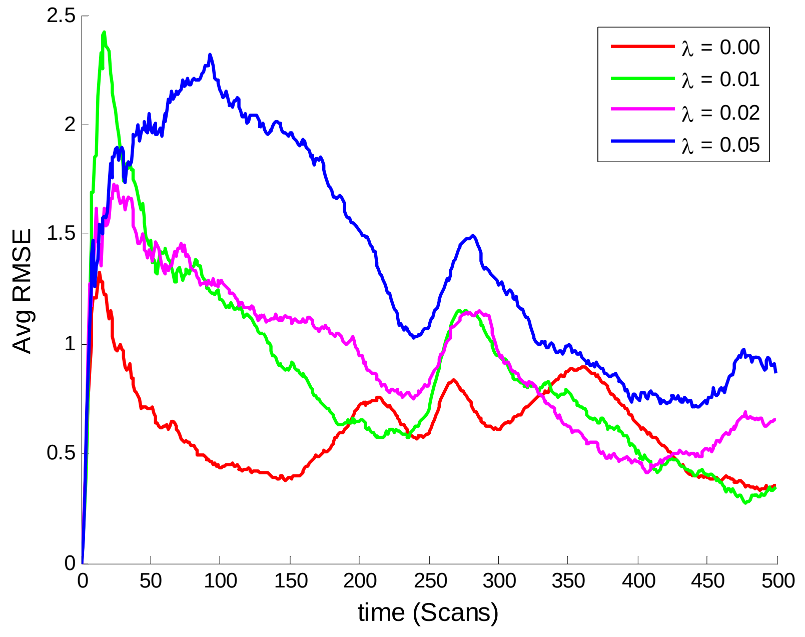

Figure 15. Average RMSE for Algorithm I: 3D LP, $\sigma_q = 0.0005$.

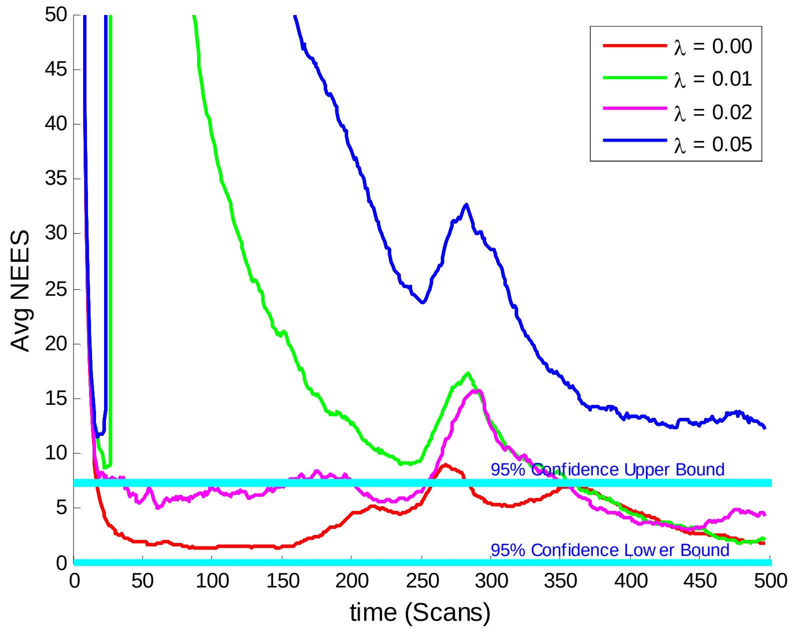

Figure 16. Average NEES for Algorithm I: 4D LP, $\sigma_q = 0.0005$.

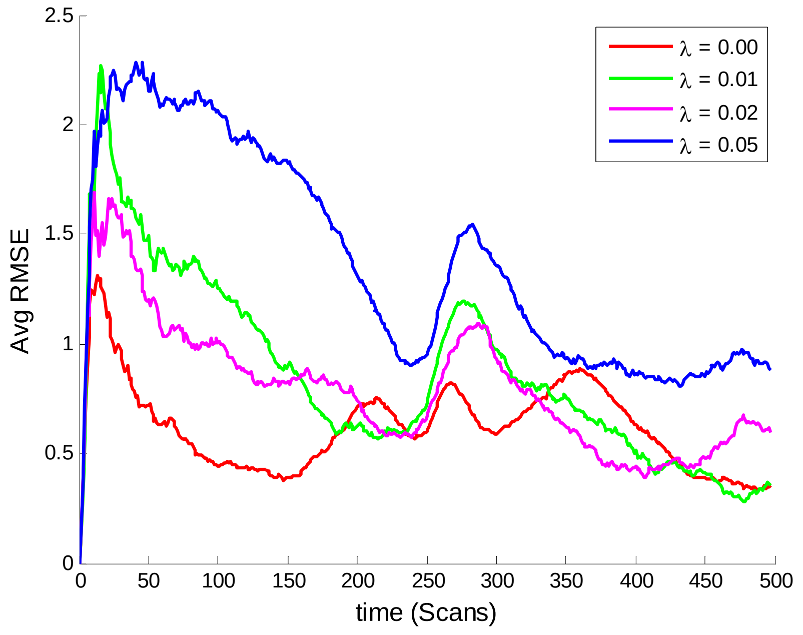

Figure 17. Average RMSE for Algorithm I: 4D LP, $\sigma_q = 0.0005$.

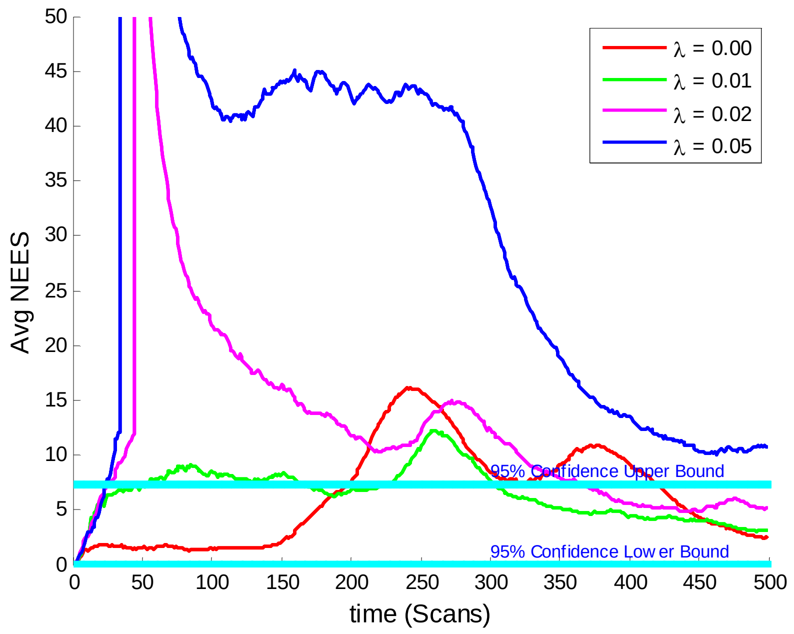

Figure 18. Average NEES for Algorithm II: 2D-3D LP, $\sigma_q = 0.0005$.

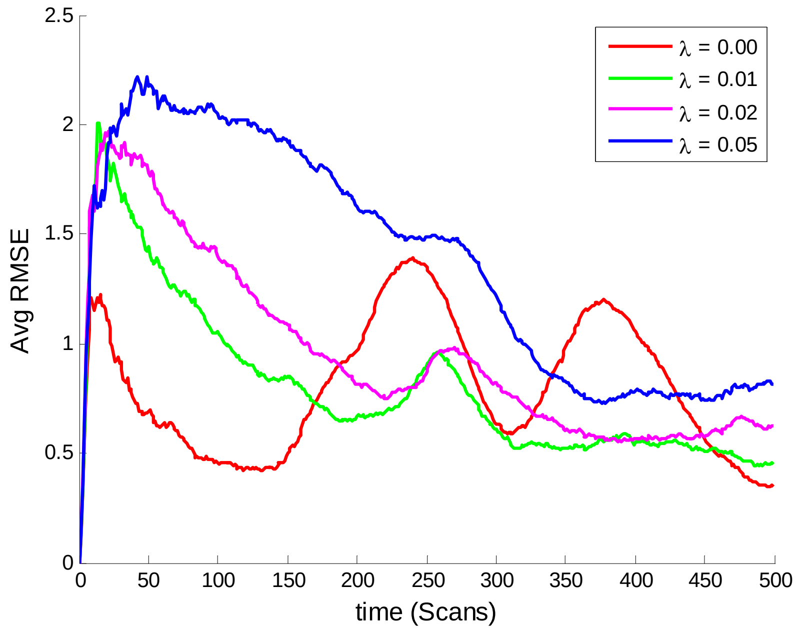

Figure 19. Average RMSE for Algorithm II: 2D-3D LP, $\sigma_q = 0.0005$.

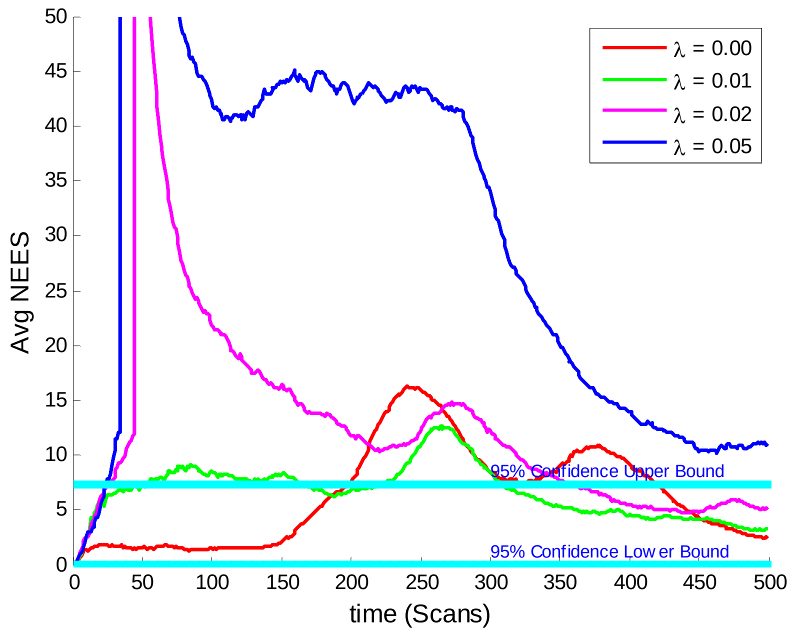

Figure 20. Average NEES for Algorithm II: 2D-4D LP, $\sigma_q = 0.0005$.

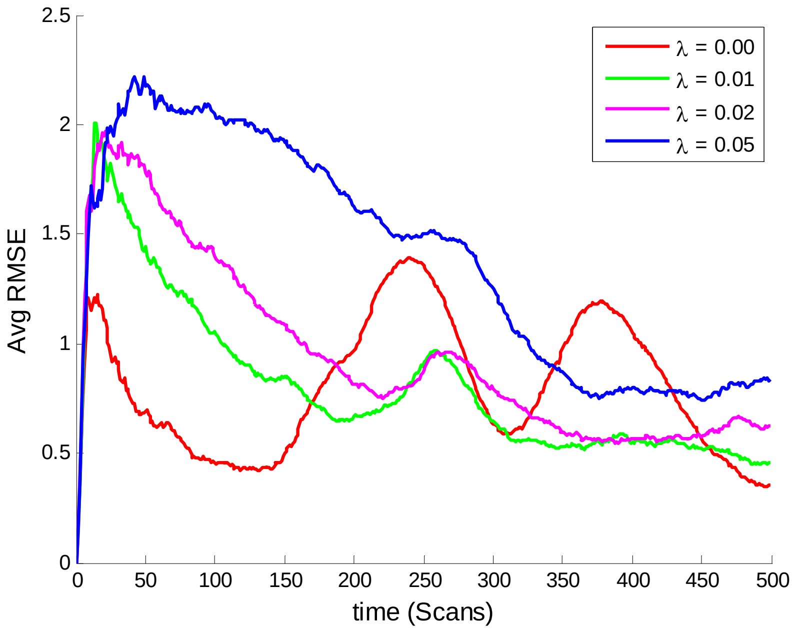

Figure 21. Average RMSE for Algorithm II: 2D-4D LP, $\sigma_q = 0.0005$.

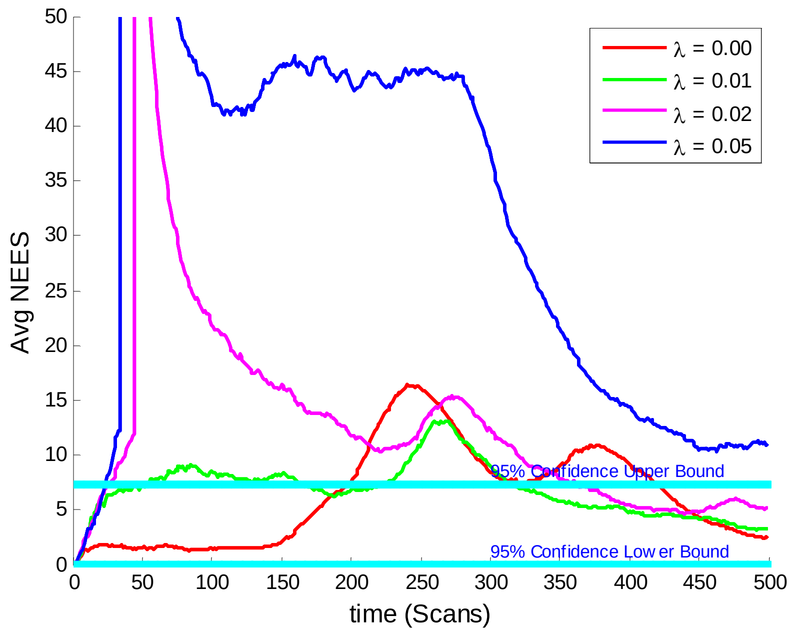

Figure 22. Average NEES for Algorithm II: 2D-6D LP, $\sigma_q = 0.0005$.

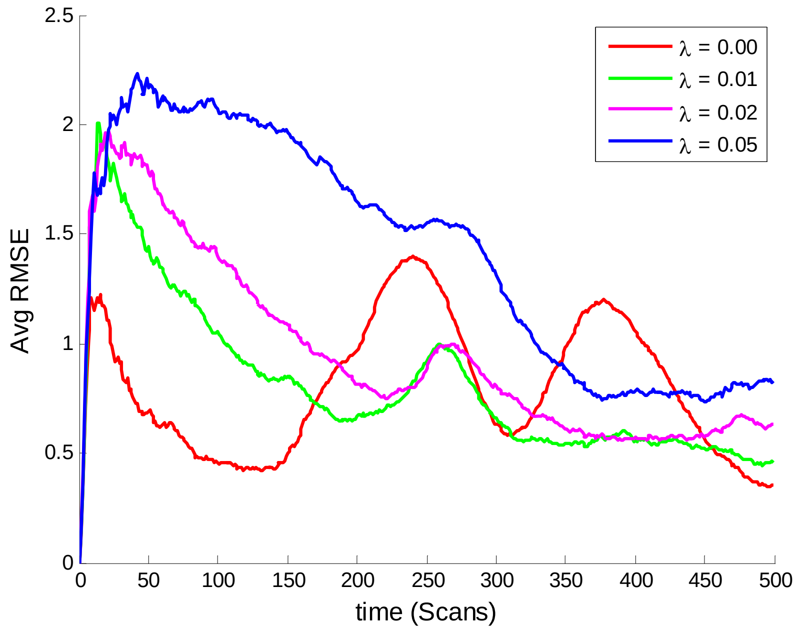

Figure 23. Average RMSE for Algorithm II: 2D-6D LP, $\sigma_q = 0.0005$.

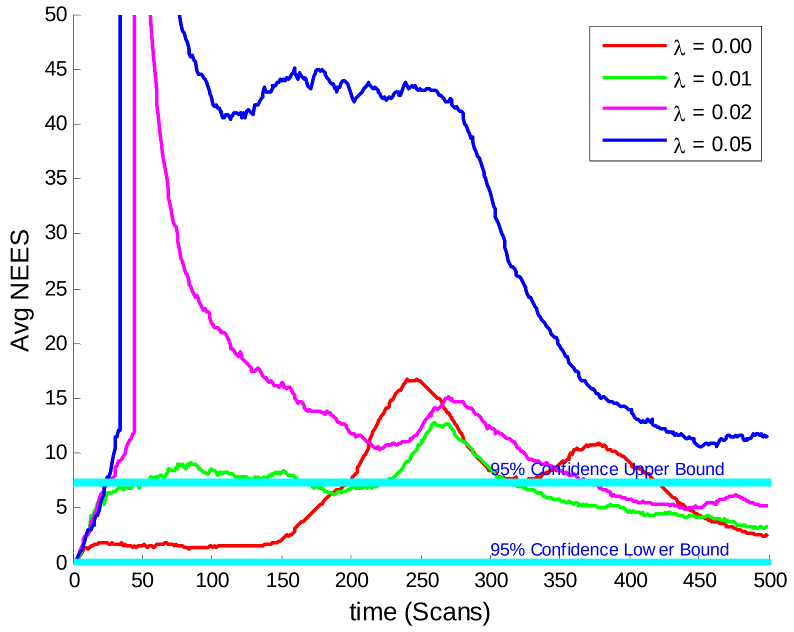

Figure 24. Average NEES for Algorithm II: 2D-11D LP, $\sigma_q = 0.0005$.

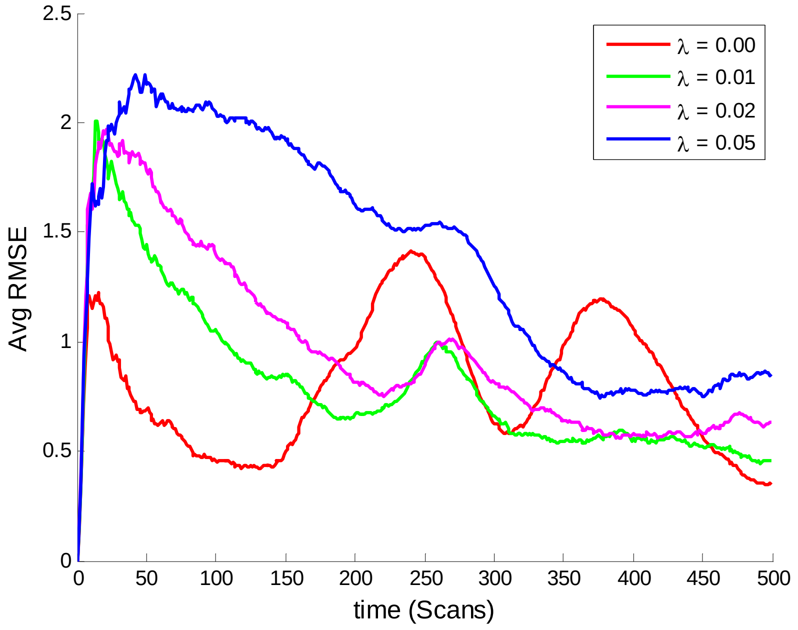

Figure 25. Average RMSE for Algorithm II: 2D-11D LP, $\sigma_q = 0.0005$.

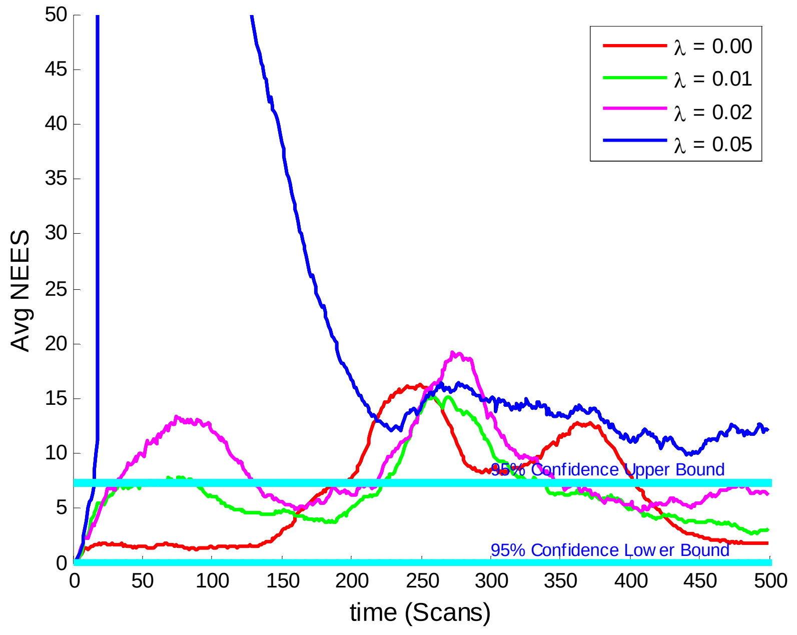

Figure 26. Average NEES for Algorithm II: 2D-3D LP, $\sigma_q = 0.002$.

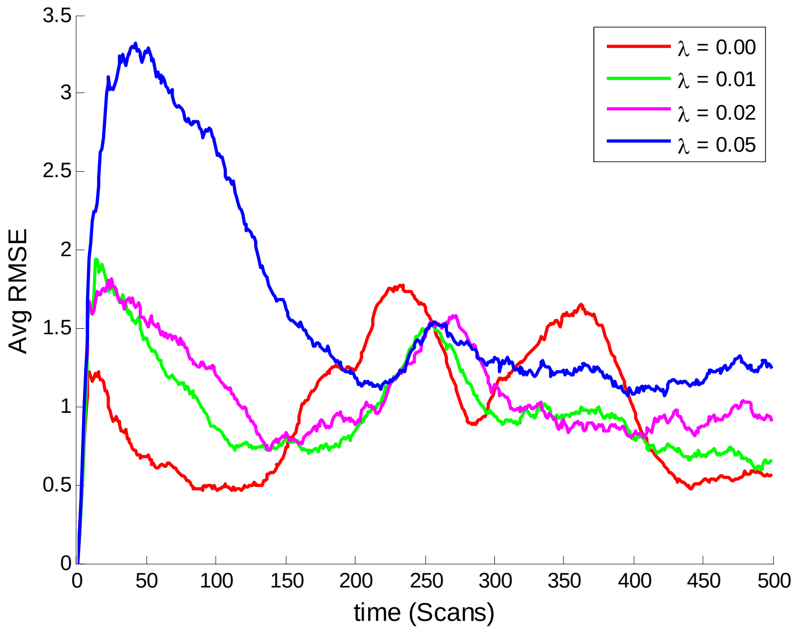

Figure 27. Average RMSE for Algorithm II: 2D-3D LP, $\sigma_q = 0.002$.

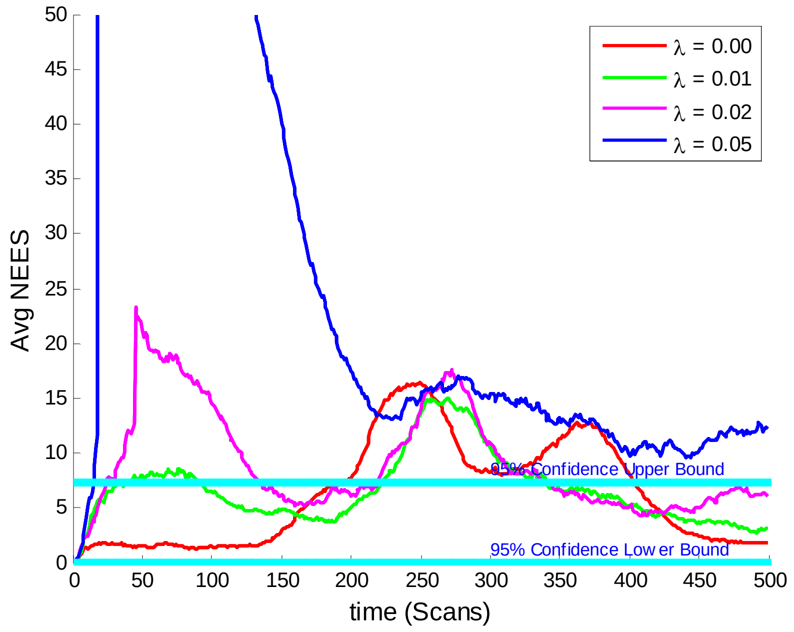

Figure 28. Average NEES for Algorithm II: 2D-4D LP, $\sigma_q = 0.002$.

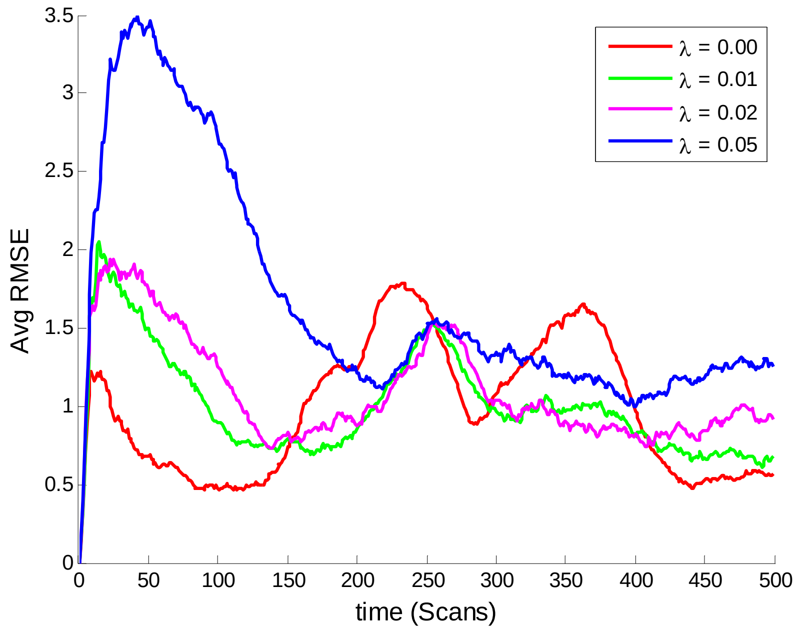

Figure 29. Average RMSE for Algorithm II: 2D-4D LP, $\sigma_q = 0.002$.

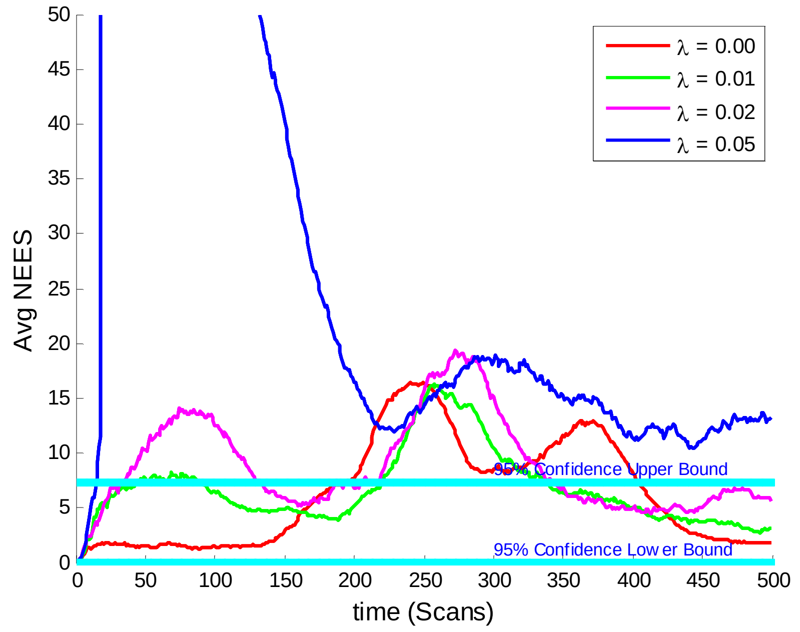

Figure 30. Average NEES for Algorithm II: 2D-6D LP, $\sigma_q = 0.002$.

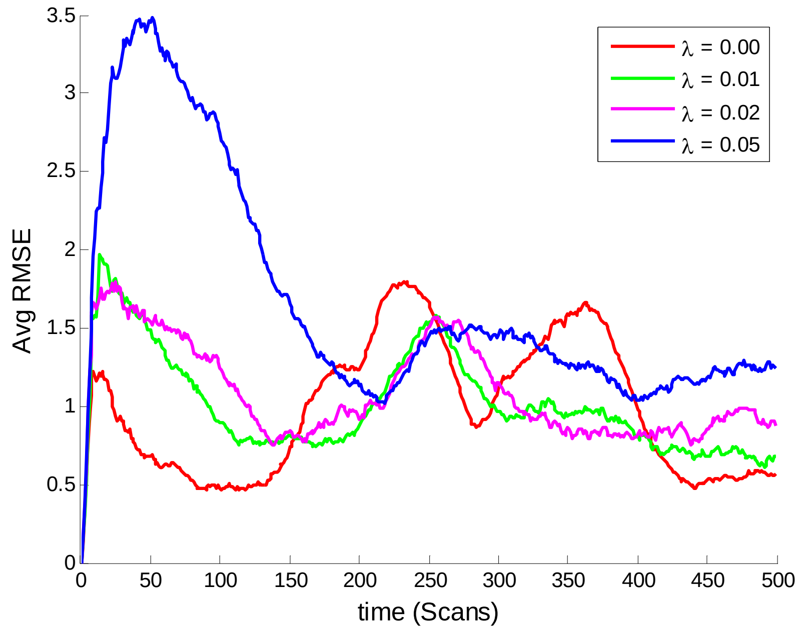

Figure 31. Average RMSE for Algorithm II: 2D-6D LP, $\sigma_q = 0.002$.

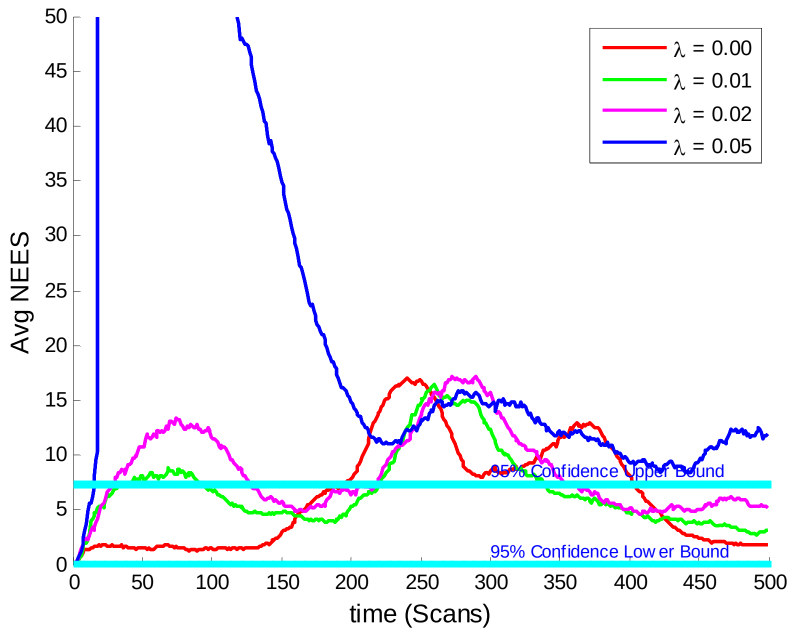

Figure 32. Average NEES for Algorithm II: 2D-11D LP, $\sigma_q = 0.002$.

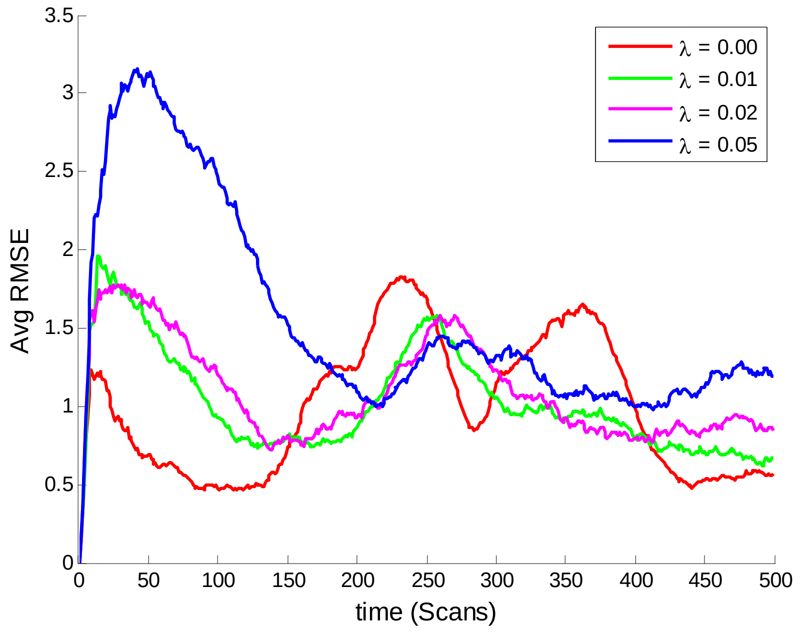

Figure 33. Average RMSE for Algorithm II: 2D-11D LP, $\sigma_q = 0.002$.

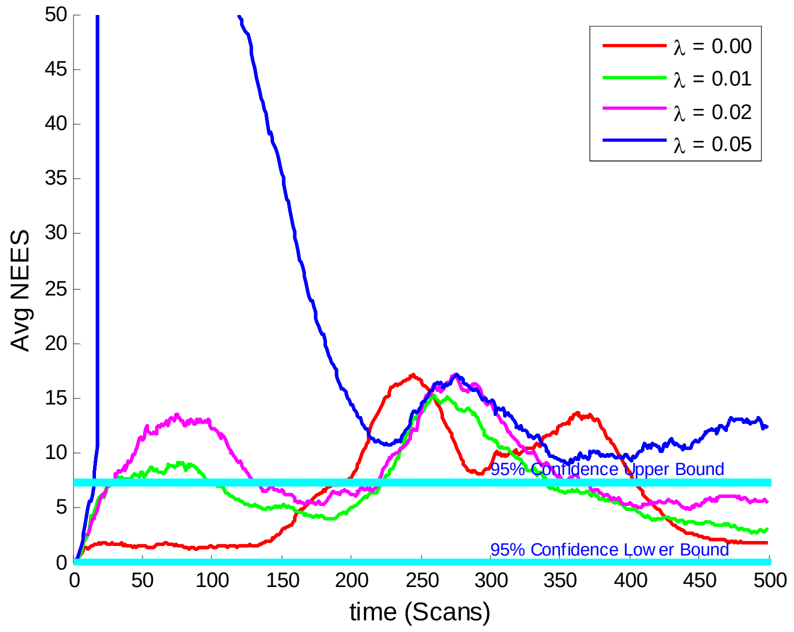

Figure 34. Average NEES for Algorithm II: 2D-16D LP, $\sigma_q = 0.002$.

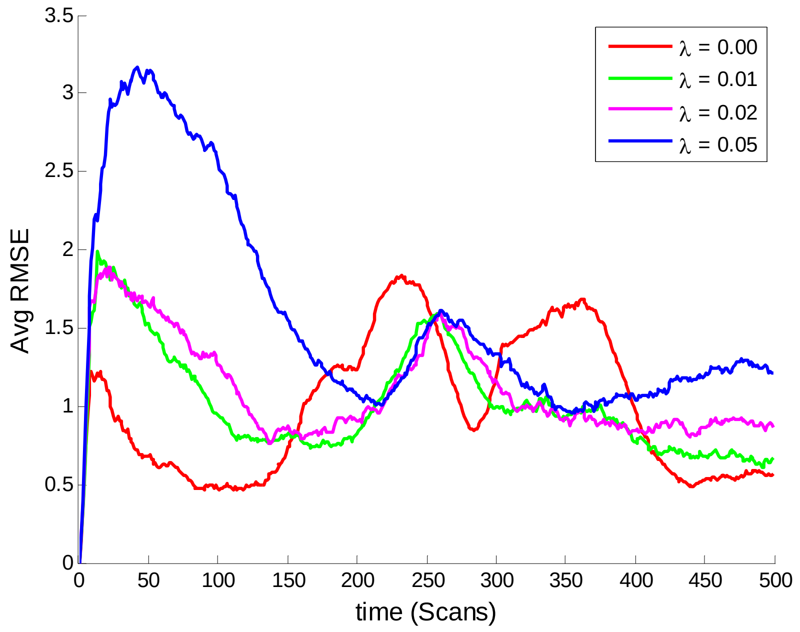

Figure 35. Average RMSE for Algorithm II: 2D-16D LP, $\sigma_q = 0.002$.

LIST OF REFERENCES

[1] P. Willett, T. E. Luginbuhl, and E. Giannopoulos, "MHT tracking for crossing sonar targets," presented at the SPIE Conference on Signal and Data Processing of Small Targets, San Diego, CA, Aug. 2007, Paper No. 6699–47.

[2] J. Areta, Y. Bar-Shalom, M. Levedahl, and K. R. Pattipati, "Hierarchical track association and fusion for a networked surveillance system," *J. of Advances in Information Fusion*, vol. 1, no. 2, Dec. 2006.

[3] M. Berkelaar, K. Eikland, and P. Notebaert, "Introduction to lp_solve 5.5.2.0," 2004. Available: http://lpsolve.sourceforge.net/5.5/.

[4] V. J. Aidala and S. E. Hammel, "Utilization of modified polar coordinates for bearings-only tracking," *IEEE Trans. Automatic Control*, vol. AC-28, no. 3, Mar. 1983.

[5] S. C. Nardone, A. G. Lindgren, and K. F. Gong, "Fundamental properties and performance of conventional bearings-only target motion analysis," *IEEE Trans. Automatic Control*, vol. AC-29, no. 9, Sep. 1984.

[6] K. C. Ho and Y. T. Chan, "An asymptotically unbiased estimator for bearings-only and Doppler-bearing target motion analysis," *IEEE Trans. Signal Processing*, vol. 54, no. 3, Mar. 2006.

[7] R. J. Urick, *Principles of Underwater Sound*, 3rd ed. New York: McGraw Hill, 1983, pp. 54–64.

[8] L. J. Ziomek, *Fundamentals of Acoustic Field Theory and Space-Time Signal Processing*. Boca Raton, FL: CRC Press, Inc., 1995, pp. 589–594.

[9] R. L. Streit and T. E. Luginbuhl, "Probabilistic Multi-Hypothesis Tracking," Naval Undersea Warfare Center Division, Newport, RI, NUWC-NPT Tech. Rep. 10,428, 15 Feb. 1995.

[10] D. T. Dunham and R. G. Hutchins, "Tracking multiple targets in cluttered environments with a probabilistic multi-hypothesis tracker," in *Proc. of SPIE Annual International Symposium on AeroSense*, Orlando, FL, vol. 3086, pp. 284-295, Apr. 1997.

[11] D. Reid, "An algorithm for tracking multiple targets," *IEEE Trans. Automatic Control*, vol. AC-24, no. 6, Dec. 1979.

[12] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Norwood, MA: Artech House, 1999, pp. 360–369.

[13] Y. Bar-Shalom and X. R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. Storrs, CT: YBS Publishing, 1995.

[14] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*. San Diego, CA: Academic Press, Inc., 1998, pp. 222–237.

[15] O. E. Drummond, “Multiple target tracking with multiple frame, probabilistic data association,” in *Proc. of SPIE Signal and Data Processing of Small Targets 1993*, Orlando, FL, vol. 1954, pp. 394–409, Apr. 1993.

[16] J. A. Roecker, “Multiple scan joint probabilistic data association,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 31, no. 3, Jul. 1995.

[17] S. Puranik and J. K. Tugnait, “Tracking of multiple maneuvering targets using multiscan JPDA and IMM filtering,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 1, Jan. 2007.

[18] R. J. Fitzgerald, “Track biases and coalescence with the probabilistic data association,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-21, pp. 822–825, Nov. 1985.

[19] K. R. Pattipati, S. Deb, Y. Bar-Shalom, and R. B. Washburn, Jr, “A new relaxation algorithm and passive sensor data association,” *IEEE Trans. Automatic Control*, vol. 37, no. 2, Feb. 1992.

[20] A. B. Poore and N. Rijavec, “A Lagrangian relaxation algorithm for multi-dimensional assignment problems arising from multi-target tracking,” *SIAM J. of Optimization*, vol. 3, no. 3, pp. 645–663, Aug 1993.

[21] S. Deb, M. Yeddanapudi, K. Pattipati, and Y. Bar-Shalom, “A generalized S-D assignment algorithm for multisensor-multitarget state estimation,” *IEEE Trans. on Aerospace and Electronic Systems*, vol. 33, no. 2, Apr. 1997.

[22] X. Li, Z. Luo, K. M.Wong, and E.Bosse, “An interior point linear programming approach to two-scan data association,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 35, no. 2, Apr. 1999.

[23] C. L. Morefield, “Application of 0-1 Integer Programming to Multitarget Tracking Problems, *IEEE Trans. Automatic Control*, vol. AC-22, pp. 302–311, Jun. 1977.

[24] H. Chen, T. Kirubarajan, and Y. Bar-Shalom, “Tracking of spawning targets with multiple finite resolution sensors,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 44, no. 1, Jan. 2008.

[25] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York, NY: John Wiley & Sons, Inc., 2001.

[26] Y. Bar-Shalom, S. S. Blackman, and R. J. Fitzgerald, “Dimensionless score function for multiple hypothesis tracking,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 43, no. 1, Jan. 2007.

[27] M. R. Chummun, T. Kirubarajan, K. R. Pattipati, and Y.,Bar-Shalom, “Fast data association using multidimensional assignment with clustering,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 3, Jul. 2001.

[28] Wang. H., Kirubarajan. T., Bar-Shalom, Y., “Precision Large Scale Air Traffic Surveillance using IMM/Assignment estimators,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 37, no. 3, Jan. 1999.

[29] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. San Diego, CA: Academic Press, 1982.

[30] G. B. Dantzig, *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press, 1963.

[31] S. I. Gass, *Linear Programming Methods and Applications*, 5th ed. New York: McGraw-Hill, 1985.

[32] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York: John Wiley & Sons, Inc., 1988.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Daphne Kapolka
   Naval Postgraduate School
   Monterey, California

4. Professor Robert G. Hutchins
   Naval Postgraduate School
   Monterey, California

5. Commander
   Naval Undersea Warfare Center Division, Newport
   Attn: Tod E. Luginbuhl
   Newport, Rhode Island

6. Commander
   Naval Undersea Warfare Center Division, Newport
   Attn: Sunil Mathews (Code 1511)
   Newport, Rhode Island

7. Commander
   Naval Undersea Warfare Center Division, Newport
   Attn: Technical Library (Code 1144)
   Newport, Rhode Island